Decoding

Administrivia

- Form a 1- or 2- person group (TAs can help) ASAP!
- Initial reading list due Tuesday, 9/10
  - one-sentence statement of scope + 5 papers
  - you can change anything later
  - latex + bibtex + templates on website → pdf
- Assignment 1 due Tuesday 9/17
  - On website

Lecture Outline

1. Viterbi algorithm
2. Decoding more generally
3. Five views

Hidden Markov Model

- A model over sequences of symbols, but there is missing information associated with each symbol: its "state."
  - Assume a finite set of possible states, Λ.

$$p(\text{start}, s_1, w_1, s_2, w_2, \ldots, s_n, w_n \text{stop}) = \prod_{i=1}^{n+1} \eta(w_i \mid s_i) \times \gamma(s_i \mid s_{i-1})$$

- A *joint* model over the observable symbols and their hidden/latent/unknown classes.

Key Algorithms for HMMs

Given the HMM and a sequence:

1. The most probable state sequence?
2. The probability of the word sequence?
3. The probability distribution over states, for each word?
4. Minimum risk sequence

Given states and sequences, or just states:

5. The parameters of the HMM ($\boldsymbol{\gamma}$ and $\boldsymbol{\eta}$)?

Problem 1:
Most Likely State Sequence

- Input: HMM (**γ** and **η**) and symbol sequence **w**.
- Output: $\arg\max_{\boldsymbol{s}} p(\boldsymbol{s} \mid \boldsymbol{w}, \gamma, \eta)$

- Statistics view: maximum *a posteriori* inference
- Computational view: discrete, combinatorial optimization

Example

I	suspect	the	present	forecast	is	pessimistic	.
CD	JJ	DT	JJ	NN	NNS	JJ	.
NN	NN	JJ	NN	VB	VBZ		
NNP	VB	NN	RB	VBD			
PRP	VBP	NNP	VB	VBN			
		VBP	VBP	VBP			
4	4	5	5	5	2	1	1

4,000 possible state sequences!

Naïve Solutions

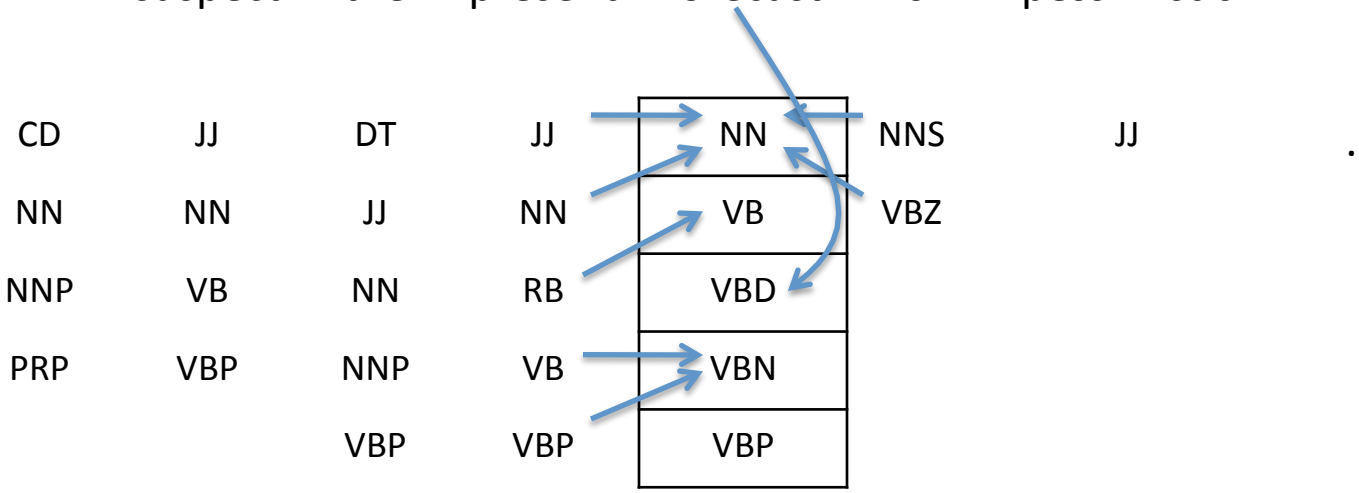
- List all the possibilities in Λ^n.
  - Correct.
  - Inefficient.
- Work left to right and greedily pick the best s_i at each point, based on s_{i-1} and w_i.
  - Not correct; solution may not be equal to:

    $$\arg\max_{\boldsymbol{s}} p(\boldsymbol{s} \mid \boldsymbol{w}, \boldsymbol{\gamma}, \boldsymbol{\eta})$$

  - But fast!

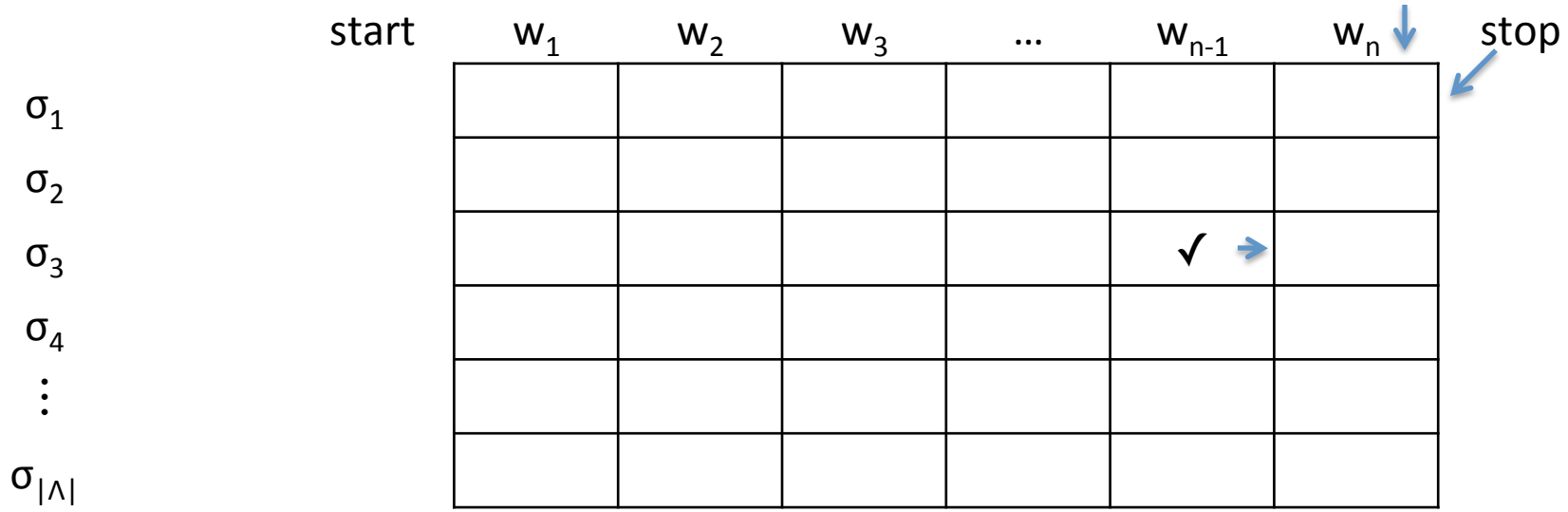
Interactions

- Each word's label depends on the word, and nearby labels.
- But given *adjacent* labels, others do not matter.



(arrows show *most preferred* label by each neighbor)

Base Case: Last Label



$$\text{score}_n(\sigma) \quad = \quad \gamma(\text{stop} \mid \sigma) \times \eta(w_n \mid \sigma) \times \gamma(\sigma \mid s_{n-1})$$

Of course, we do not actually know s_{n-1}!

Recurrence

- If I knew the score of every sequence $s_1 \ldots s_{n-1}$, I could reason easily about s_n.
  - But my decision about s_n would only depend on s_{n-1}!
- So I really only need to know the score of the best sequence ending in each s_{n-1}.
- Think of that as some "precalculation" that happens before I think about s_n.

Recurrence

- Assume we have the scores for all prefixes of the current state sequence.
  - One score for each possible last-state of the prefix.

$$\text{score}_n(\sigma) \quad = \quad \gamma(\text{stop} \mid \sigma) \times \eta(w_n \mid \sigma) \times \max_{\sigma'} \gamma(\sigma \mid \sigma') \times \text{score}_{n-1}(\sigma')$$

Recurrence

- The recurrence "bottoms out" at start.
- This leads to a simple algorithm for calculating all the scores.

$$
\begin{aligned}
\text{score}_n(\sigma) &= \gamma(\text{stop} \mid \sigma) \times \eta(w_n \mid \sigma) \times \max_{\sigma'} \gamma(\sigma \mid \sigma') \times \text{score}_{n-1}(\sigma') \\
\text{score}_{n-1}(\sigma) &= \eta(w_{n-1} \mid \sigma) \times \max_{\sigma'} \gamma(\sigma \mid \sigma') \times \text{score}_{n-2}(\sigma') \\
\text{score}_{n-2}(\sigma) &= \eta(w_{n-2} \mid \sigma) \times \max_{\sigma'} \gamma(\sigma \mid \sigma') \times \text{score}_{n-3}(\sigma') \\
&\vdots \qquad\qquad \vdots \\
\text{score}_1(\sigma) &= \eta(w_1 \mid \sigma) \times \gamma(\sigma \mid \text{start})
\end{aligned}
$$

Viterbi Algorithm (Scores Only)

- For every σ in Λ, let:

$$\text{score}_1(\sigma) \quad = \quad \eta(w_1 \mid \sigma) \times \gamma(\sigma \mid \text{start})$$

- For i = 2 to n – 1, for every σ in Λ:

$$\text{score}_i(\sigma) \quad = \quad \eta(w_i \mid \sigma) \times \max_{\sigma' \in \Lambda} \gamma(\sigma \mid \sigma') \times \text{score}_{i-1}(\sigma')$$

- For every σ in Λ:

$$\text{score}_n(\sigma) \quad = \quad \gamma(\text{stop} \mid \sigma) \times \eta(w_n \mid \sigma) \times \max_{\sigma' \in \Lambda} \gamma(\sigma \mid \sigma') \times \text{score}_{n-1}(\sigma')$$

- Claim:

$$\max_{\boldsymbol{s}} p(\boldsymbol{s}, \boldsymbol{w} \mid \boldsymbol{\gamma}, \boldsymbol{\eta}) = \max_{\sigma \in \Lambda} \text{score}_n(\sigma)$$

Exploiting Distributivity

$$
\begin{aligned}
\max_{\sigma\in\Lambda} \text{score}_n(\sigma) &= \max_{\sigma\in\Lambda} \gamma(\text{stop} \mid \sigma) \times \eta(w_n \mid \sigma) \times \max_{\sigma'\in\Lambda} \gamma(\sigma \mid \sigma') \times \text{score}_{n-1}(\sigma') \\
&= \max_{\sigma\in\Lambda} \gamma(\text{stop} \mid \sigma) \times \eta(w_n \mid \sigma) \times \max_{\sigma'\in\Lambda} \gamma(\sigma \mid \sigma') \\
&\quad \times \eta(w_{n-1} \mid \sigma') \times \max_{\sigma''\in\Lambda} \gamma(\sigma' \mid \sigma'') \times \text{score}_{n-2}(\sigma'') \\
&= \max_{\sigma\in\Lambda} \gamma(\text{stop} \mid \sigma) \times \eta(w_n \mid \sigma) \times \max_{\sigma'\in\Lambda} \gamma(\sigma \mid \sigma') \\
&\quad \times \eta(w_{n-1} \mid \sigma') \times \max_{\sigma''\in\Lambda} \gamma(\sigma' \mid \sigma'') \\
&\quad \times \eta(w_{n-2} \mid \sigma'') \times \max_{\sigma'''\in\Lambda} \gamma(\sigma'' \mid \sigma''') \times \text{score}_{n-3}(\sigma''') \\
&= \max_{\sigma,\sigma',\sigma'',\sigma'''} \gamma(\text{stop} \mid \sigma) \times \eta(w_n \mid \sigma) \times \gamma(\sigma \mid \sigma') \\
&\quad \times \eta(w_{n-1} \mid \sigma') \times \gamma(\sigma' \mid \sigma'') \\
&\quad \times \eta(w_{n-2} \mid \sigma'') \times \gamma(\sigma'' \mid \sigma''') \times \text{score}_{n-3}(\sigma''') \\
&= \max_{\boldsymbol{s}\in\Lambda^n} \prod_{i=1}^{n+1} \gamma(s_i \mid s_{i-1}) \times \eta(w_i \mid s_i)
\end{aligned}
$$

$$
\max_{\boldsymbol{s}} p(\boldsymbol{s}, \boldsymbol{w} \mid \boldsymbol{\gamma}, \boldsymbol{\eta}) = \max_{\sigma\in\Lambda} \text{score}_n(\sigma)
$$

	I	suspect	the	present	forecast	is	pessimistic	.
CD	3E-7							
DT			3E-8					
JJ		1E-9	1E-12	3E-12			7E-23	
NN	4E-6	2E-10	1E-13	6E-13	4E-16			
NNP	1E-5		4E-13					
NNS						1E-21		
PRP	4E-3							
RB				2E-14				
VB		6E-9		3E-15	2E-19			
VBD					6E-18			
VBN					4E-18			
VBP		5E-7	4E-14	4E-15	9E-19			
VBZ						6E-18		
.								2E-24
	1	2	3	4	5	6	7	8

Not Quite There

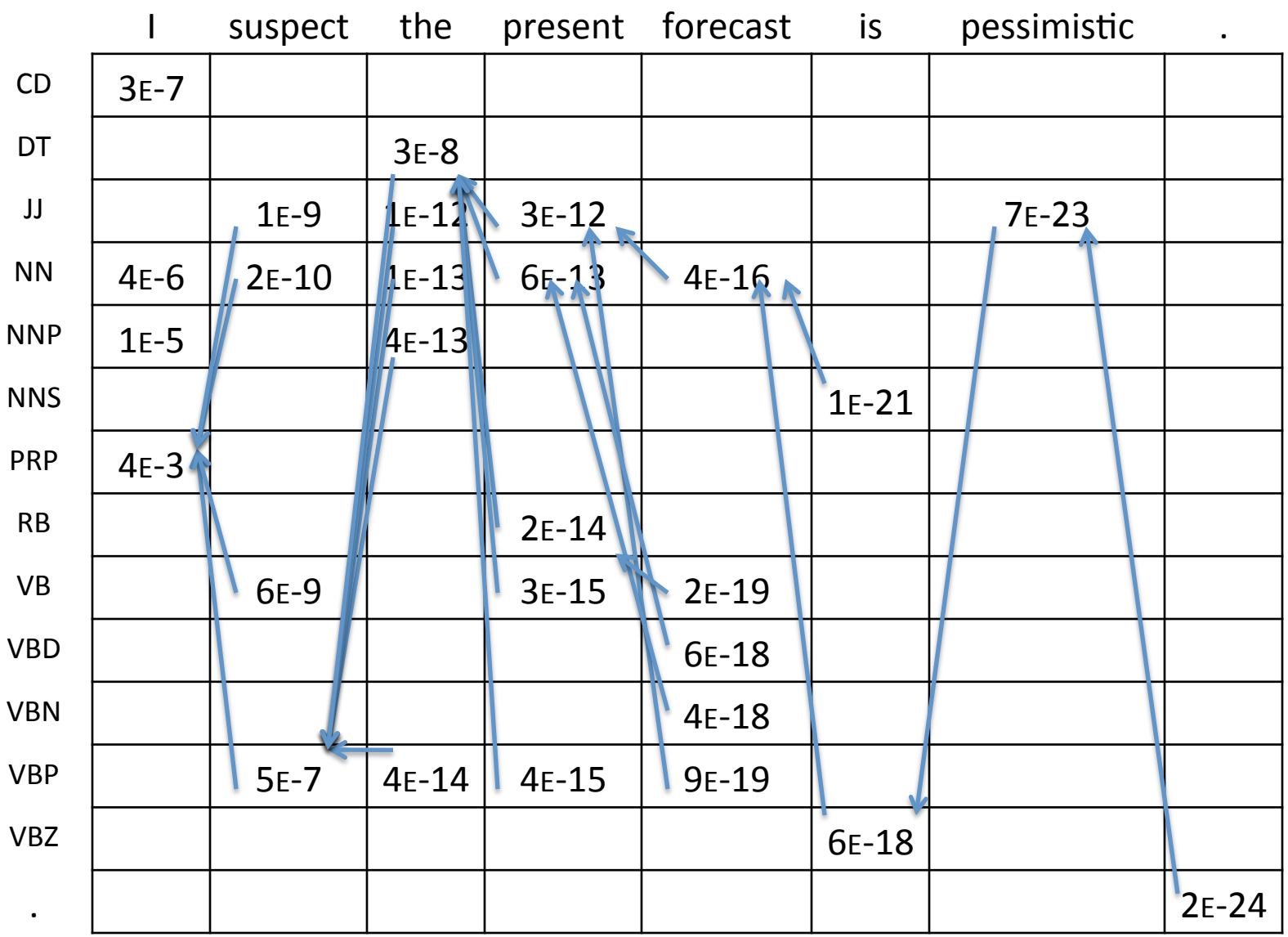
- As described, this algorithm only lets us calculate the *probability* of the best label sequence.
- It does not recover the best sequence!

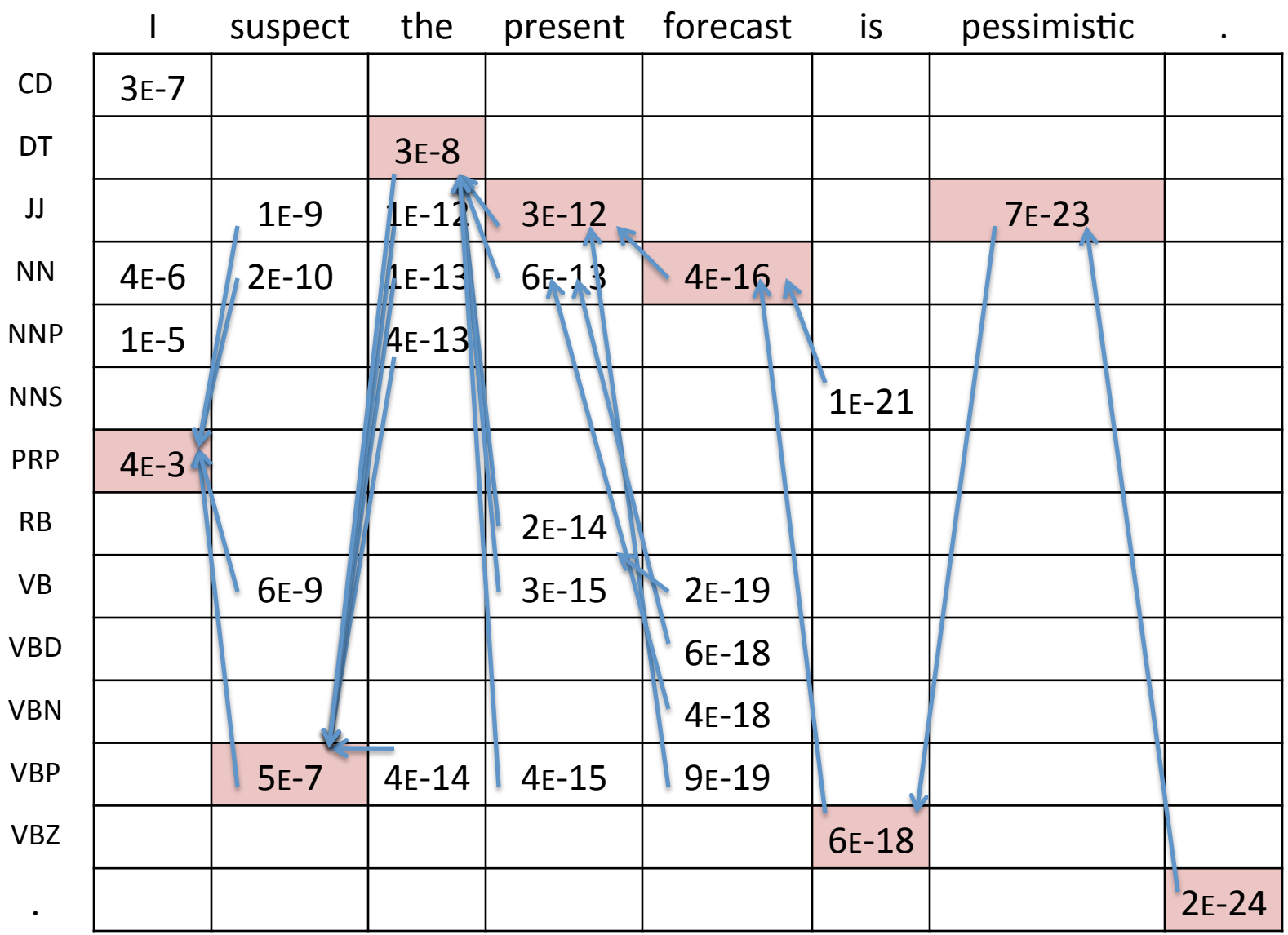
Understanding the Scores

- $\text{score}_i(\sigma)$ is the score of the best sequence labeling up through w_i, ignoring what comes later.

$$\text{score}_i(\sigma) = \max_{s_1,\ldots,s_{i-1}} p(s_1, w_1, s_2, w_2, \ldots, s_i = \sigma, w_i)$$

- Similar trick as before: if I know what s_{i+1} is, then I can use the scores to choose s_i.
- Solution: keep backpointers.


I suspect the present forecast is pessimistic .
CD
DT
JJ
NN
NNP
NNS
PRP
RB
VB
VBD
VBN
VBP
VBZ
.
3E-7
3E-8
1E-9
1E-12
3E-12
7E-23
4E-6
2E-10
1E-13
6E-13
4E-16
1E-5
4E-13
1E-21
4E-3
2E-14
6E-9
3E-15
2E-19
6E-18
4E-18
5E-7
4E-14
4E-15
9E-19
6E-18
2E-24


I suspect the present forecast is pessimistic .
CD
DT
JJ
NN
NNP
NNS
PRP
RB
VB
VBD
VBN
VBP
VBZ
.
3E-7
3E-8
1E-9
1E-12
3E-12
7E-23
4E-6
2E-10
1E-13
6E-13
4E-16
1E-5
4E-13
1E-21
4E-3
2E-14
6E-9
3E-15
2E-19
6E-18
4E-18
5E-7
4E-14
4E-15
9E-19
6E-18
2E-24

Viterbi Algorithm

- For every σ in Λ, let:

$$\text{score}_1(\sigma) \quad = \quad \eta(w_1 \mid \sigma) \times \gamma(\sigma \mid \text{start})$$

- For i = 2 to n – 1, for every σ in Λ:

$$\text{score}_i(\sigma) \quad = \quad \eta(w_i \mid \sigma) \times \max_{\sigma' \in \Lambda} \gamma(\sigma \mid \sigma') \times \text{score}_{i-1}(\sigma')$$

$$\text{bp}_i(\sigma) = \arg\max_{\sigma' \in \Lambda} \gamma(\sigma \mid \sigma') \times \text{score}_{i-1}(\sigma')$$

- For every σ in Λ:

$$\text{score}_n(\sigma) \quad = \quad \gamma(\text{stop} \mid \sigma) \times \eta(w_n \mid \sigma) \times \max_{\sigma' \in \Lambda} \gamma(\sigma \mid \sigma') \times \text{score}_{n-1}(\sigma')$$

$$\text{bp}_n(\sigma) = \arg\max_{\sigma' \in \Lambda} \gamma(\sigma \mid \sigma') \times \text{score}_{n-1}(\sigma')$$

Viterbi Algorithm: Backtrace

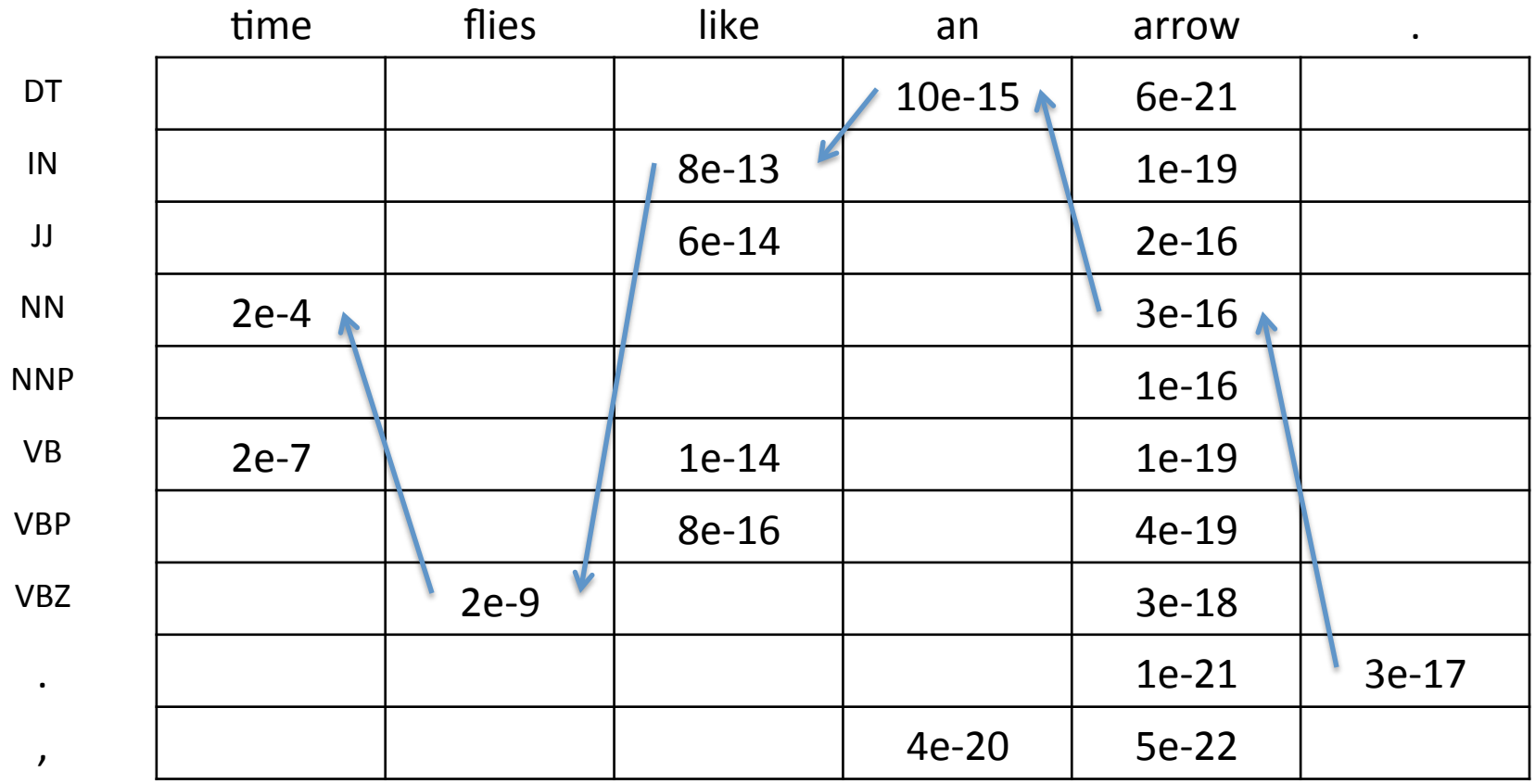
- After calculating all score and bp values, start by choosing s_n to maximize $score_n$.
- Then let $s_{n-1} = bp_n(s_n)$.

- In general, $s_{i-1} = bp_i(s_i)$.

Another Example

	time	flies	like	an	arrow	.
DT				10e-15	6e-21	
IN			8e-13		1e-19	
JJ			6e-14		2e-16	
NN	2e-4				3e-16	
NNP					1e-16	
VB	2e-7		1e-14		1e-19	
VBP			8e-16		4e-19	
VBZ		2e-9			3e-18	
.					1e-21	3e-17
,				4e-20	5e-22	

Another Example



Lecture Outline

✓ Viterbi algorithm

2. Decoding more generally
3. Five views

Inference

- Eventually, you need to run your structured predictor on test data!
- For sequence labeling and segmentation models with very local interactions, decoding is usually accomplished by something “like” Viterbi algorithm.

Random Variables

- A variable whose value depends on chance
- Denoted by capital letters: X, Y, Z
- Associated with sets of possible values: Val(X)
- A single possible value: $x \in \text{Val}(X)$
- Probabilistic modeling: defining distributions over r.v.s

- There's more than one way to map your structured prediction problem to random variables!

Probabilistic Inference Problems

Given values for some random variables ($\mathbf{X} \subset \mathbf{V}$) …

- *Most Probable Explanation: what are the *most probable* values of the *rest* of the r.v.s $\mathbf{V} \setminus \mathbf{X}$?

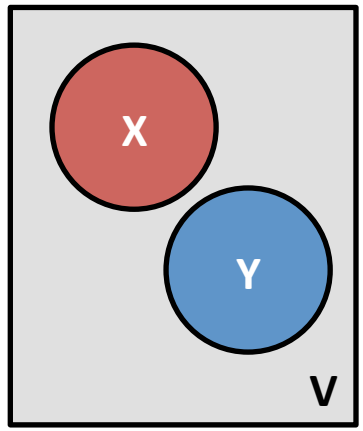
(More generally …)

- *Maximum *A Posteriori* (MAP): what are the most probable values of *some* other r.v.s, $\mathbf{Y} \subset (\mathbf{V} \setminus \mathbf{X})$?

These do not need to be probabilistic! Change "most probable" to "maximum scoring."

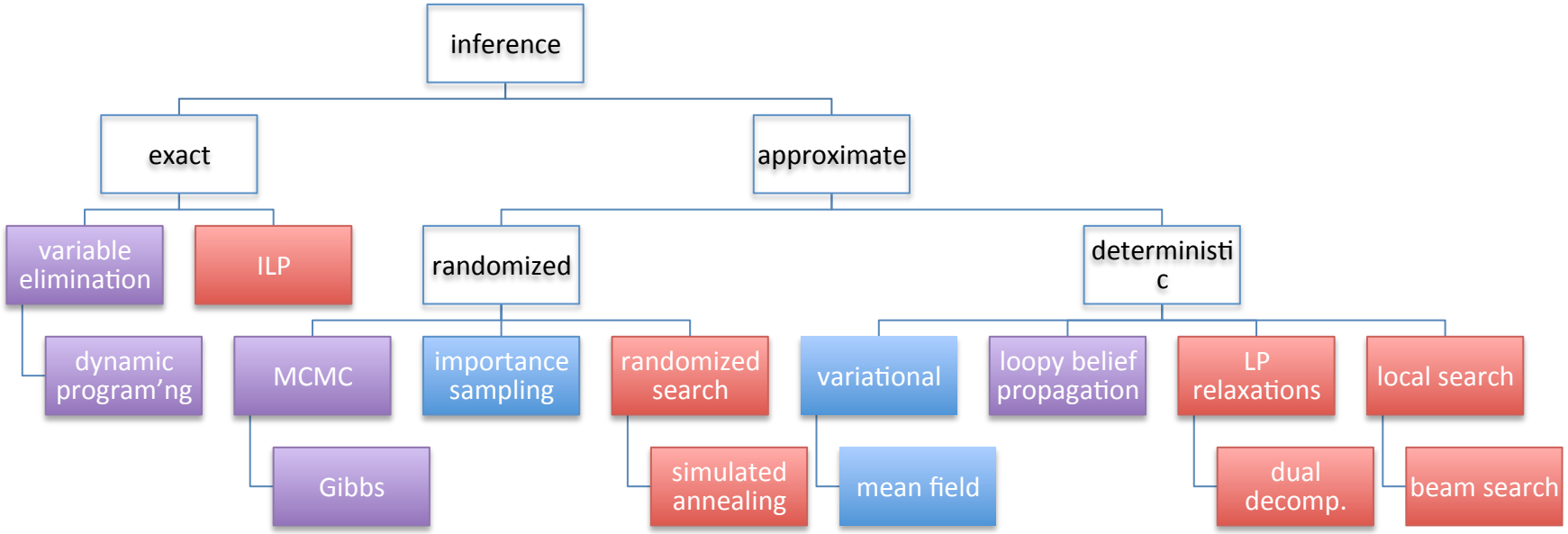
- Random sampling from the posterior over values of **Y**
- Full posterior over values of **Y**
- Marginal probabilities from the posterior over **Y**

- *Minimum Bayes risk: What is the **Y** with the lowest expected cost?
- *Cost-augmented decoding: What is the most *dangerous* value of **Y**, compared to true **y***?



*Different kinds of **decoding**.

Approaches to Inference



red = hard inference
blue = soft inference
purple = both

Hidden Markov Model

- **X** and **Y** are both sequences of symbols
  - **X** is a sequence from the vocabulary Σ
  - **Y** is a sequence from the state space Λ

$$p(\boldsymbol{Y} = \boldsymbol{s}, \boldsymbol{X} = \boldsymbol{w}) =$$
$$p(\text{start}, s_1, w_1, s_2, w_2, \ldots, s_n, w_n \text{stop}) = \prod_{i=1}^{n+1} \eta(w_i \mid s_i) \times \gamma(s_i \mid s_{i-1})$$

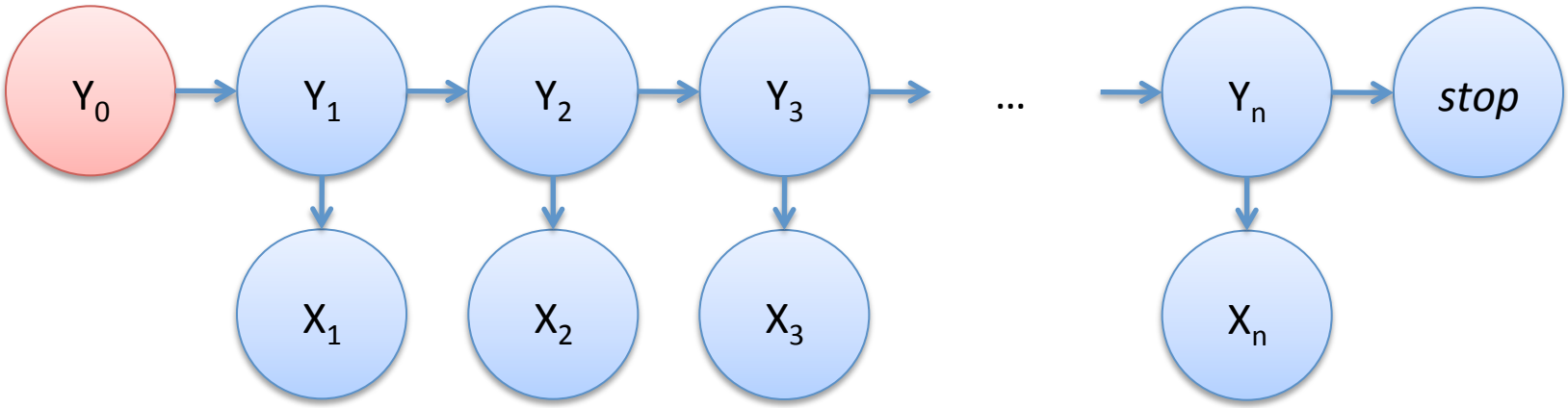
- Parameters:
  - Transitions **γ** including γ(*stop* | s), γ(s | *start*)
  - Emissions **η**

Hidden Markov Model

- The joint model's independence assumptions are easy to capture with a Bayesian network.

$$p(\boldsymbol{Y} = \boldsymbol{s}, \boldsymbol{X} = \boldsymbol{w}) =$$

$$p(\text{start}, s_1, w_1, s_2, w_2, \ldots, s_n, w_n \text{stop}) = \prod_{i=1}^{n+1} \eta(w_i \mid s_i) \times \gamma(s_i \mid s_{i-1})$$

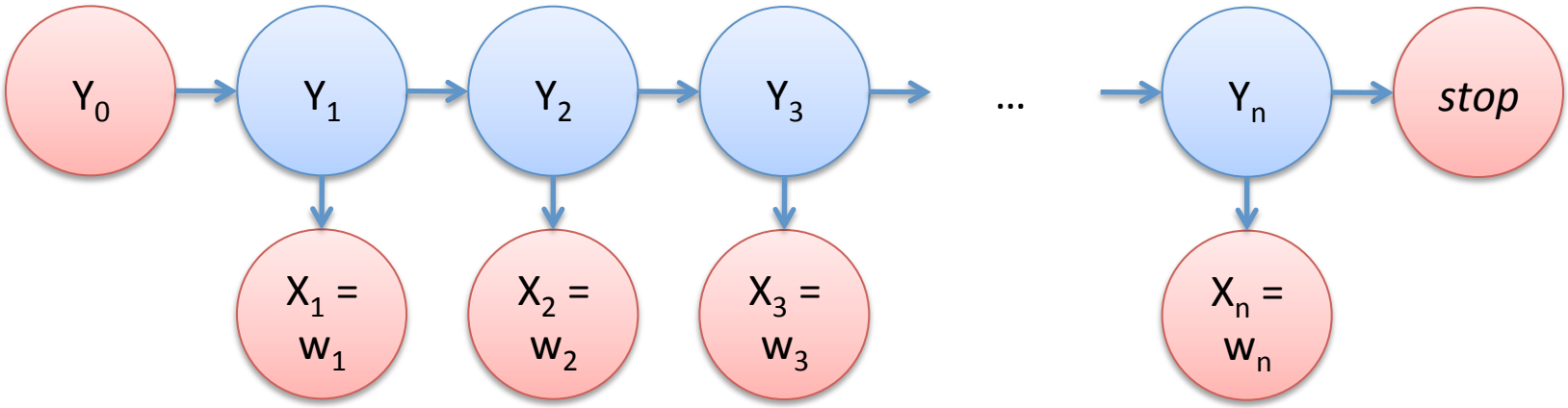


Hidden Markov Model

- The MPE/MAP inference problem is to find the most probable value of **Y** given **X** = **x**.

$$p(\boldsymbol{Y} = \boldsymbol{s}, \boldsymbol{X} = \boldsymbol{w}) =$$

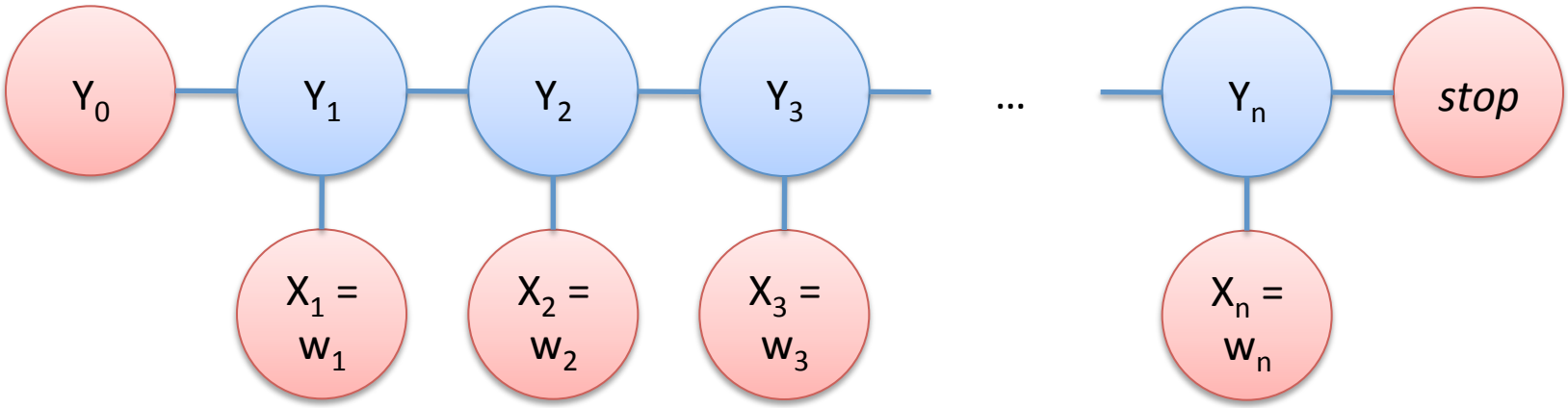
$$p(\text{start}, s_1, w_1, s_2, w_2, \ldots, s_n, w_n \text{stop}) \quad = \quad \prod_{i=1}^{n+1} \eta(w_i \mid s_i) \times \gamma(s_i \mid s_{i-1})$$



Hidden Markov Model

- The MPE/MAP inference problem is to find the most probable value of **Y** given **X** = **x**.

- Markov network:



Markov Network

- A different graphical model representation; undirected. Vertices are still r.v.s.
- Every clique C in the graph gets a *local* scoring function ϕ_C that maps assignments to values.

$$mulscore(\boldsymbol{x}, \boldsymbol{y}) = \prod_{C \in \mathcal{C}} \phi_C(\Pi_C(\boldsymbol{x}, \boldsymbol{y}))$$

$$addscore(\boldsymbol{x}, \boldsymbol{y}) = \sum_{C \in \mathcal{C}} \log \phi_C(\Pi_C(\boldsymbol{x}, \boldsymbol{y}))$$

- This score can be *globally* renormalized to obtain a probabilistic interpretation. (Not today.)

Restriction #1

1. The score function needs to *factor locally*.
   – The more locally, the better!

$$score(\boldsymbol{x}, \boldsymbol{y}) = \sum_{C \in \mathcal{C}} \log \phi_C(\Pi_C(\boldsymbol{x}, \boldsymbol{y}))$$

Linear Models

- Define a feature vector function **g** that maps (**x**, **y**) pairs into d-dimensional real space.
- Score is linear in **g**(**x**, **y**).

$$\begin{aligned} score(\boldsymbol{x}, \boldsymbol{y}) &= \mathbf{w}^\top \mathbf{g}(\boldsymbol{x}, \boldsymbol{y}) \\ \boldsymbol{y}^* &= \arg\max_{\boldsymbol{y} \in \mathcal{Y}_{\boldsymbol{x}}} \mathbf{w}^\top \mathbf{g}(\boldsymbol{x}, \boldsymbol{y}) \end{aligned}$$

- Results:
  - decoding seeks **y** to maximize the score.
  - learning seeks **w** to … do something we'll talk about later.
- Extremely general!

Generic Noisy Channel as Linear Model

$$
\begin{aligned}
\hat{\boldsymbol{y}} &= \arg\max_{\boldsymbol{y}} \log\left(p(\boldsymbol{y}) \cdot p(\boldsymbol{x} \mid \boldsymbol{y})\right) \\
&= \arg\max_{\boldsymbol{y}} \log p(\boldsymbol{y}) + \log p(\boldsymbol{x} \mid \boldsymbol{y}) \\
&= \arg\max_{\boldsymbol{y}} w_{\boldsymbol{y}} + w_{\boldsymbol{x}|\boldsymbol{y}} \\
&= \arg\max_{\boldsymbol{y}} \mathbf{w}^\top \mathbf{g}(\boldsymbol{x}, \boldsymbol{y})
\end{aligned}
$$

- Of course, the two probability terms are typically composed of “smaller” factors; each can be understood as an exponentiated weight.

Max Ent Models as Linear Models

$$
\begin{aligned}
\hat{\boldsymbol{y}} &= \arg\max_{\boldsymbol{y}} \log p(\boldsymbol{y} \mid \boldsymbol{x}) \\
&= \arg\max_{\boldsymbol{y}} \log \frac{\exp \mathbf{w}^\top \mathbf{g}(\boldsymbol{x}, \boldsymbol{y})}{z(\boldsymbol{x})} \\
&= \arg\max_{\boldsymbol{y}} \mathbf{w}^\top \mathbf{g}(\boldsymbol{x}, \boldsymbol{y}) - \log z(\boldsymbol{x}) \\
&= \arg\max_{\boldsymbol{y}} \mathbf{w}^\top \mathbf{g}(\boldsymbol{x}, \boldsymbol{y})
\end{aligned}
$$

HMMs as Linear Models

$$
\begin{aligned}
\hat{\boldsymbol{y}} &= \arg\max_{\boldsymbol{y}} \log p(\boldsymbol{x}, \boldsymbol{y}) \\
&= \arg\max_{\boldsymbol{y}} \left(\sum_{i=1}^{n} \log p(x_i \mid y_i) + \log p(y_i \mid y_{i-1}) \right) + \log p(stop \mid y_n) \\
&= \arg\max_{\boldsymbol{y}} \left(\sum_{i=1}^{n} w_{y_i \downarrow x_i} + w_{y_{i-1} \rightarrow y_i} \right) + w_{y_n \rightarrow stop} \\
&= \arg\max_{\boldsymbol{y}} \sum_{y,x} w_{y \downarrow x} freq(y \downarrow x; \boldsymbol{y}, \boldsymbol{x}) + \sum_{y,y'} w_{y \rightarrow y'} freq(y \rightarrow y'; \boldsymbol{y}) \\
&= \arg\max_{\boldsymbol{y}} \mathbf{w}^{\top} \mathbf{g}(\boldsymbol{x}, \boldsymbol{y})
\end{aligned}
$$

Restrictions #1, #2

1. The score function needs to *factor locally*.
   – The more locally, the better!

$$score(\boldsymbol{x}, \boldsymbol{y}) = \sum_{C \in \mathcal{C}} \log \phi_C(\Pi_C(\boldsymbol{x}, \boldsymbol{y}))$$

2. The local scoring functions need to be linear in features.

$$score(\boldsymbol{x}, \boldsymbol{y}) = \mathbf{w}^\top \mathbf{g}(\boldsymbol{x}, \boldsymbol{y})$$

$$score(\boldsymbol{x}, \boldsymbol{y}) = \sum_{C \in \mathcal{C}} \mathbf{w}^\top \mathbf{f}(\Pi_C(\boldsymbol{x}, \boldsymbol{y}))$$

Running Example

		1	2	3	4	5	6	7	8	9	10
$\boldsymbol{x}$	=	Britain	sent	warships	across	the	English	Channel	Monday	to	rescue
$\boldsymbol{y}$	=	B	O	O	O	O	B	I	B	O	O
$\boldsymbol{y}'$	=	O	O	O	O	O	B	I	B	O	O

11	12	13	14	15	16	17	18	19	20
Britons	stranded	by	Eyjafjallajökull	's	volcanic	ash	cloud	.	
B	O	O	B	O	O	O	O	O	O
B	O	O	B	O	O	O	O	O	O

- IOB sequence labeling, here applied to NER
- Often solved with HMMs, CRFs, M^3Ns …

feature function $g : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}$		$g(\boldsymbol{x}, \boldsymbol{y})$	$g(\boldsymbol{x}, \boldsymbol{y}')$
bias:	count of i s.t. $y_i = \mathsf{B}$	5	4
	count of i s.t. $y_i = \mathsf{I}$	1	1
	count of i s.t. $y_i = \mathsf{O}$	14	15
lexical:	count of i s.t. $x_i = \mathit{Britain}$ and $y_i = \mathsf{B}$	1	0
	count of i s.t. $x_i = \mathit{Britain}$ and $y_i = \mathsf{I}$	0	0
	count of i s.t. $x_i = \mathit{Britain}$ and $y_i = \mathsf{O}$	0	1
downcased:	count of i s.t. $\mathit{lc}(x_i) = \mathit{britain}$ and $y_i = \mathsf{B}$	1	0
	count of i s.t. $\mathit{lc}(x_i) = \mathit{britain}$ and $y_i = \mathsf{I}$	0	0
	count of i s.t. $\mathit{lc}(x_i) = \mathit{britain}$ and $y_i = \mathsf{O}$	0	1
	count of i s.t. $\mathit{lc}(x_i) = \mathit{sent}$ and $y_i = \mathsf{O}$	1	1
	count of i s.t. $\mathit{lc}(x_i) = \mathit{warships}$ and $y_i = \mathsf{O}$	1	1
shape:	count of i s.t. $\mathit{shape}(x_i) = \mathit{Aaaaaaa}$ and $y_i = \mathsf{B}$	3	2
	count of i s.t. $\mathit{shape}(x_i) = \mathit{Aaaaaaa}$ and $y_i = \mathsf{I}$	1	1
	count of i s.t. $\mathit{shape}(x_i) = \mathit{Aaaaaaa}$ and $y_i = \mathsf{O}$	0	1
prefix:	count of i s.t. $\mathit{pre}_1(x_i) = B$ and $y_i = \mathsf{B}$	2	1
	count of i s.t. $\mathit{pre}_1(x_i) = B$ and $y_i = \mathsf{I}$	0	0
	count of i s.t. $\mathit{pre}_1(x_i) = B$ and $y_i = \mathsf{O}$	0	1
	count of i s.t. $\mathit{pre}_1(x_i) = s$ and $y_i = \mathsf{O}$	2	2
	count of i s.t. $\mathit{shape}(\mathit{pre}_1(x_i)) = A$ and $y_i = \mathsf{B}$	5	4
	count of i s.t. $\mathit{shape}(\mathit{pre}_1(x_i)) = A$ and $y_i = \mathsf{I}$	1	1
	count of i s.t. $\mathit{shape}(\mathit{pre}_1(x_i)) = A$ and $y_i = \mathsf{O}$	0	1
	$[\![\mathit{shape}(\mathit{pre}_1(x_1)) = A \wedge y_1 = \mathsf{B}]\!]$	1	0
	$[\![\mathit{shape}(\mathit{pre}_1(x_1)) = A \wedge y_1 = \mathsf{O}]\!]$	0	1
gazetteer:	count of i s.t. x_i is in the gazetteer and $y_i = \mathsf{B}$	2	1
	count of i s.t. x_i is in the gazetteer and $y_i = \mathsf{I}$	0	0
	count of i s.t. x_i is in the gazetteer and $y_i = \mathsf{O}$	0	1
	count of i s.t. $x_i = \mathit{sent}$ and $y_i = \mathsf{O}$	1	1

(What is *Not* A Linear Model?)

- Probabilistic models with hidden variables, requiring general MAP inference:

$$\arg\max_{\boldsymbol{y}} p(\boldsymbol{y} \mid \boldsymbol{x}) = \arg\max_{\boldsymbol{y}} \sum_{\boldsymbol{z}} p(\boldsymbol{y}, \boldsymbol{z} \mid \boldsymbol{x})$$

- Models based on non-linear kernels

$$\arg\max_{\boldsymbol{y}} \mathbf{w}^\top \mathbf{g}(\boldsymbol{x}, \boldsymbol{y}) \quad = \quad \arg\max_{\boldsymbol{y}} \sum_{i=1}^{N} \alpha_i K\left(\langle \boldsymbol{x}_i, \boldsymbol{y}_i \rangle, \langle \boldsymbol{x}, \boldsymbol{y} \rangle\right)$$

Lecture Outline

- ✓ Viterbi algorithm
- ✓ Decoding more generally
3. Five views

1. Probabilistic Graphical Models

- View the linguistic structure as a collection of random variables that are interdependent.
- Represent interdependencies as a directed or undirected graphical model.
- Conditional probability tables (BNs) or factors (MNs) encode the probability distribution.
- Use standard techniques from PGMs to decode.

Inference in Graphical Models

- General algorithm for exact MPE inference: variable elimination.
  - Iteratively solve for the best values of each variable conditioned on values of "preceding" neighbors.
  - Then trace back.
  - Challenge: order the r.v.s for efficiency!

The Viterbi algorithm is an instance of max-product variable elimination!

MAP is Linear Decoding

- Bayesian network:

$$\sum_i \log p(x_i \mid \text{parents}(X_i)) + \sum_j \log p(y_j \mid \text{parents}(Y_j))$$

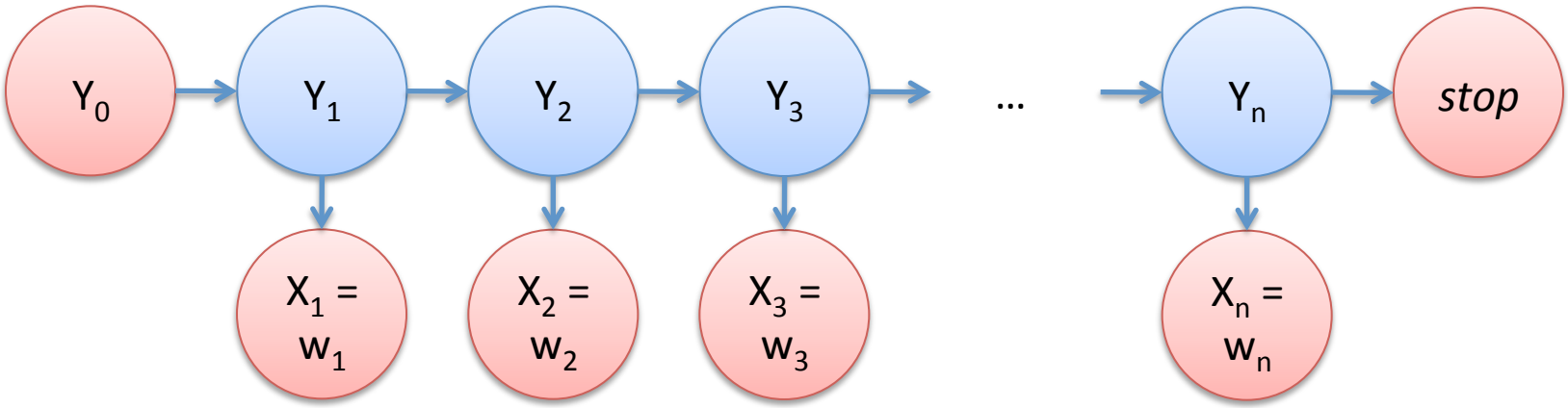
- Markov network:

$$\sum_C \log \phi_C \left(\{x_i\}_{i \in C}, \{y_j\}_{j \in C} \right)$$

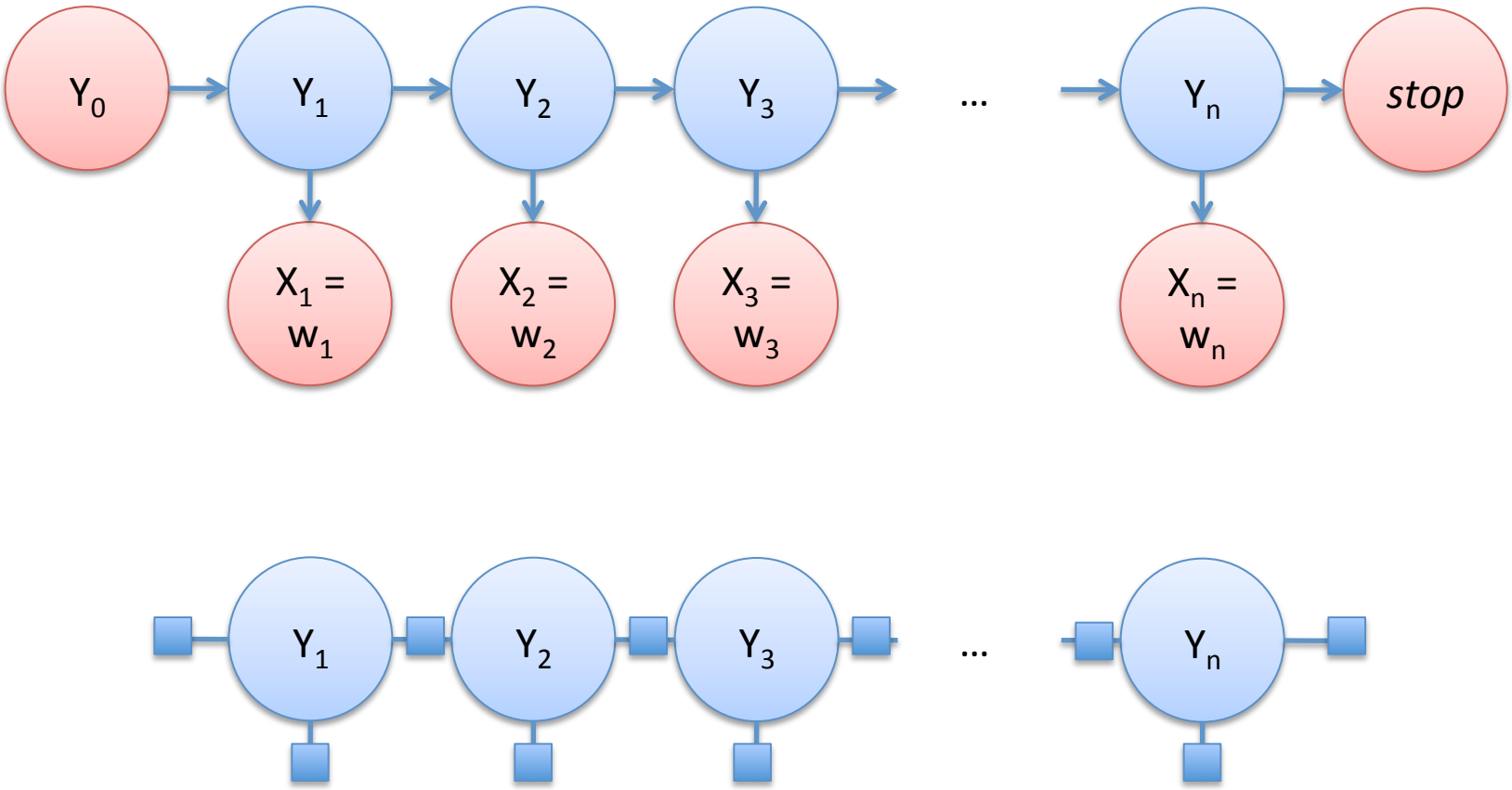
- This works if every variable is in **X** or **Y**.

Hidden Markov Model

- When we eliminate Y_1, we take a product of three relevant factors.
  - $\gamma(Y_1 \mid \textit{start})$
  - $\eta(w_1 \mid Y_1)$, reduced to the observed value w_1
  - $\gamma(Y_2 \mid Y_1)$

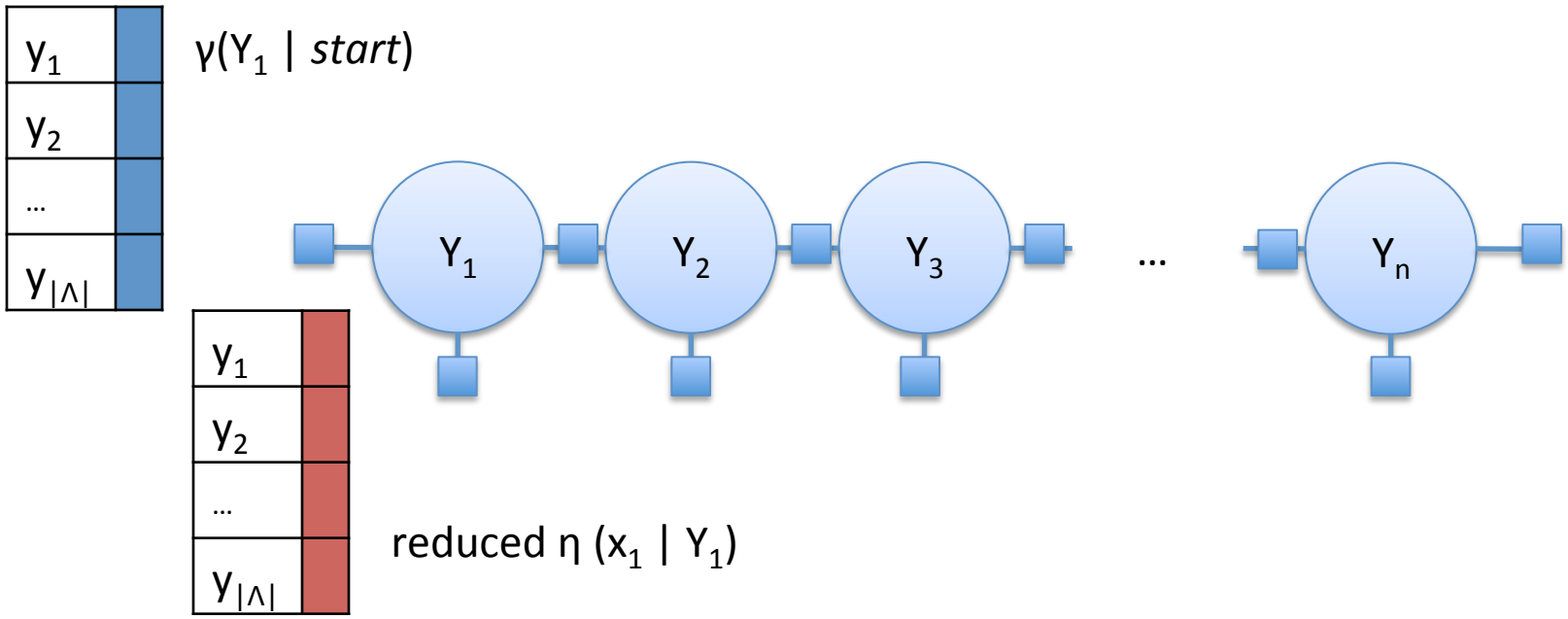


Factor Representation



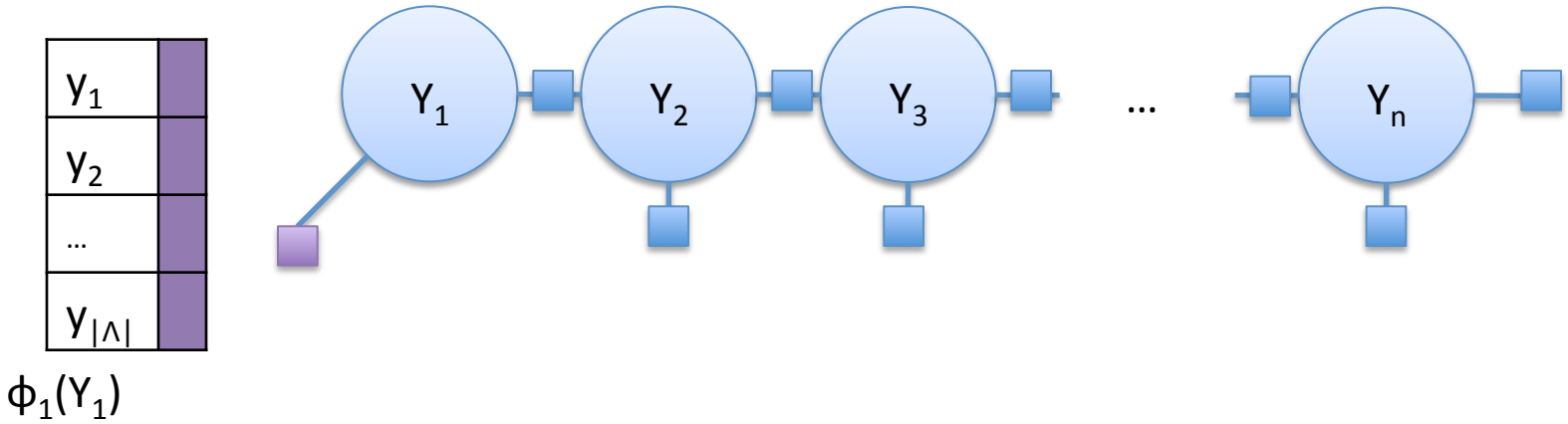
Hidden Markov Model

- When we eliminate Y_1, we first take a product of two factors that only involve Y_1.



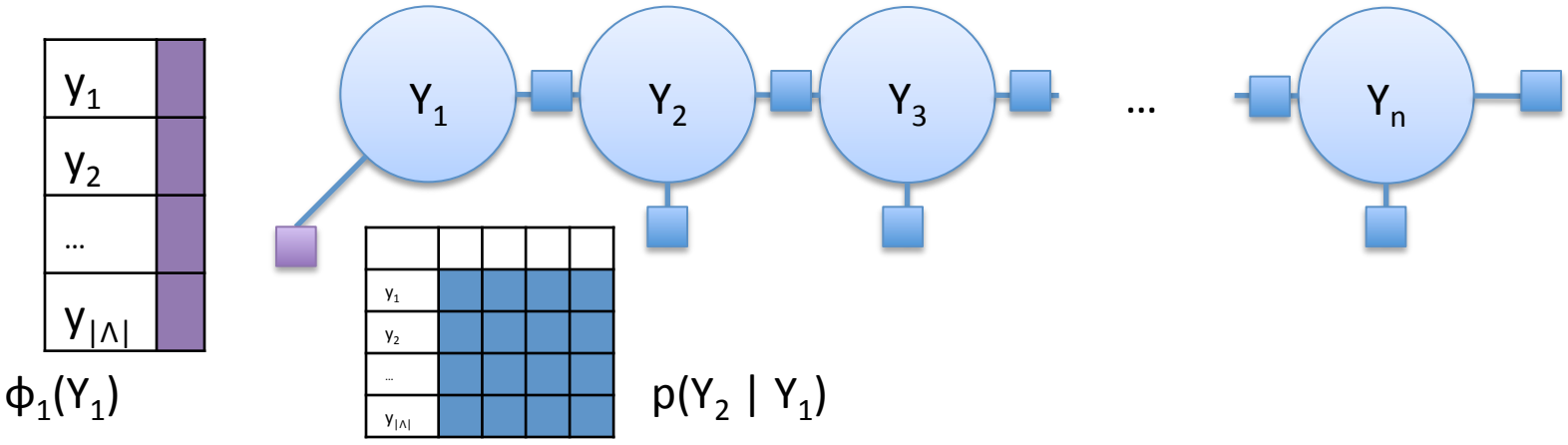
Hidden Markov Model

- When we eliminate Y_1, we first take a product of two factors that only involve Y_1.
- This is the Viterbi probability vector for Y_1.



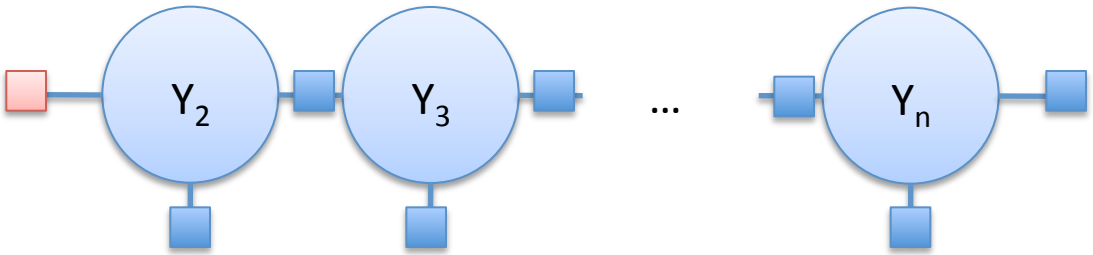
Hidden Markov Model

- When we eliminate Y_1, we first take a product of two factors that only involve Y_1.
- This is the Viterbi probability vector for Y_1.
- Eliminating Y_1 equates to solving the Viterbi probabilities for Y_2.



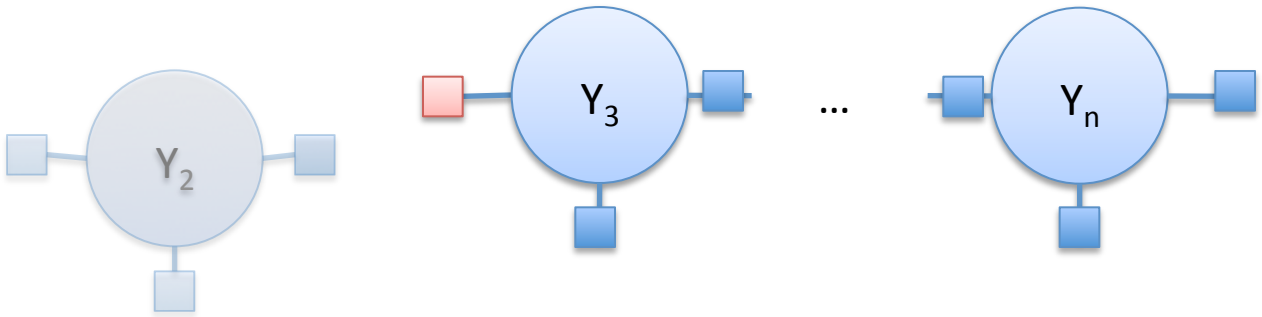
Hidden Markov Model

- Product of all factors involving Y_1, then reduce.
  - $\phi_2(Y_2) = \max_{y \in Val(Y_1)} (\phi_1(y) \times p(Y_2 \mid y))$
  - This factor holds Viterbi probabilities for Y_2.



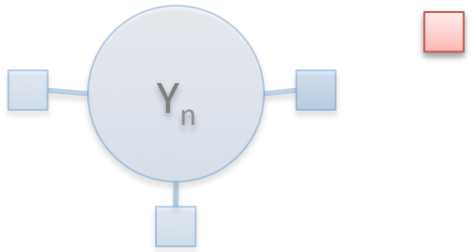
Hidden Markov Model

- When we eliminate Y_2, we take a product of the analogous two relevant factors.
- Then reduce.
  - $\phi_3(Y_3) = \max_{y \in Val(Y_2)} (\phi_2(y) \times p(Y_3 \mid y))$



Hidden Markov Model

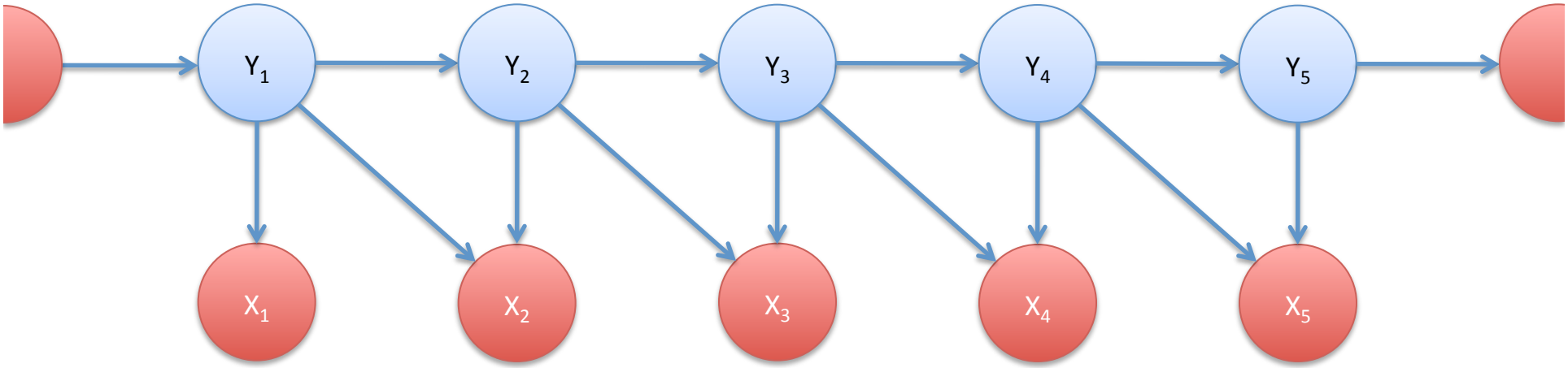
- At the end, we have one final factor with one row, ϕ_{n+1}.
- This is the score of the best sequence.
- Use backtrace to recover values.



Why Think This Way?

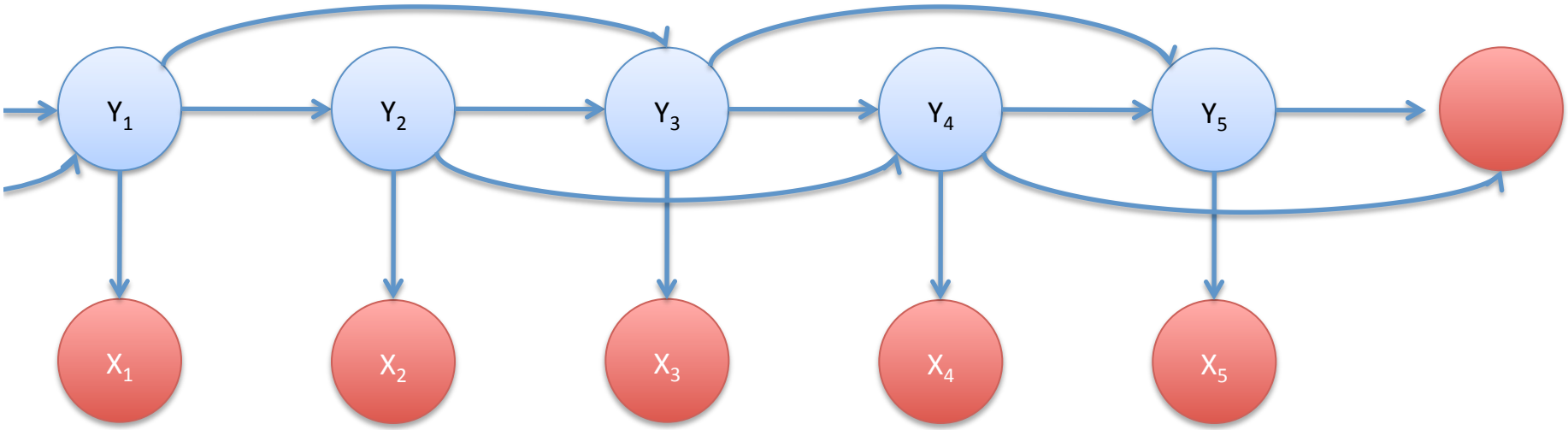
- Easy to see how to generalize HMMs.
  - More evidence
  - More factors
  - More hidden structure
  - More dependencies
- Probabilistic interpretation of factors is *not* central to finding the "best" **Y** …
  - Many factors are not conditional probability tables.

Generalization Example 1



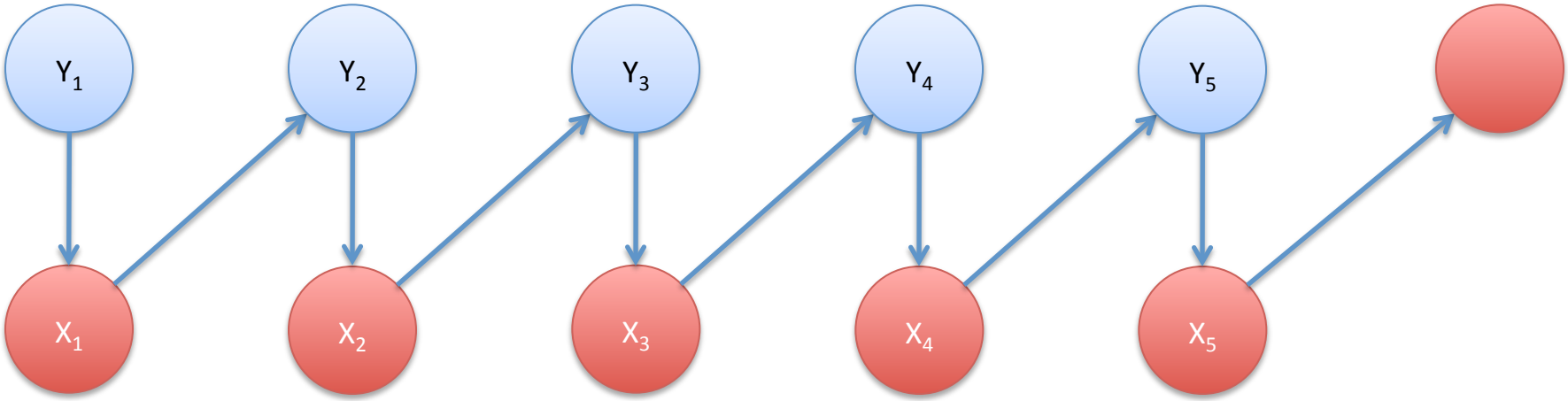
- Each word also depends on previous state.

Generalization Example 2



- "Trigram" HMM

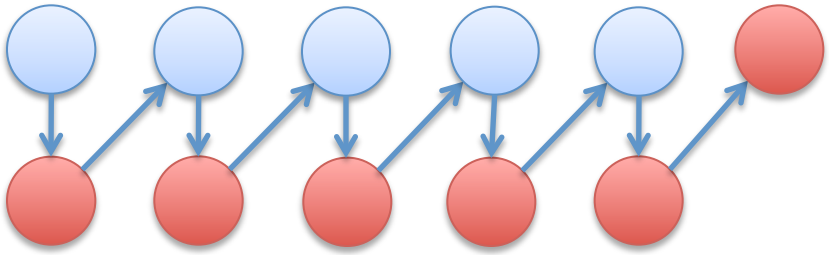
Generalization Example 3

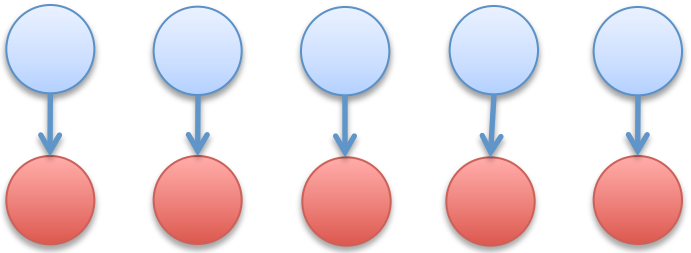


- Aggregate bigram model (Saul and Pereira, 1997)

Inference in Graphical Models

- Remember: more edges make inference more expensive.
  - Fewer edges means stronger independence.
- Really pleasant:





Inference in Graphical Models

- Remember: more edges make inference more expensive.
  - Fewer edges means stronger independence.
- Really unpleasant: