Sequence Models

Noah Smith

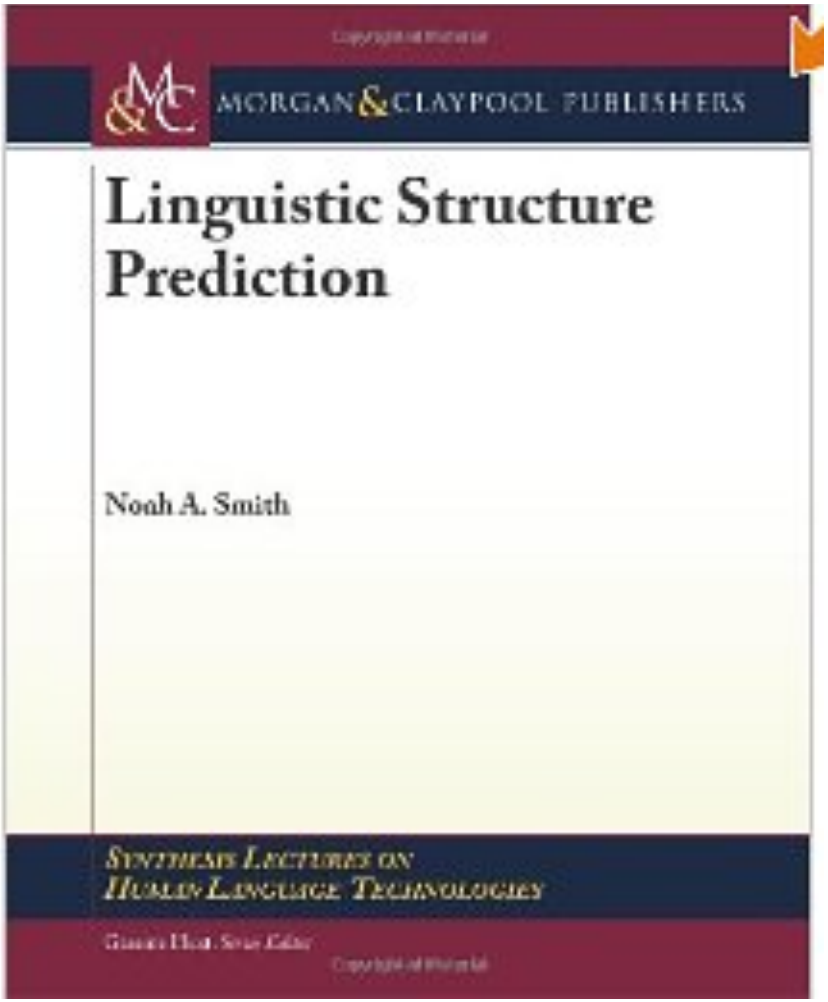
Computer Science & Engineering
University of Washington

nasmith@cs.washington.edu

Lecture Outline

1. Markov models
2. Hidden Markov models
3. Viterbi algorithm
4. Other inference algorithms for HMMs
5. Learning algorithms for HMMs

Shameless Self-Promotion



- *Linguistic Structure Prediction* (2011)
- Links material in this lecture to many related ideas, including some in other LxMLS lectures.
- Available in electronic and print form.

MARKOV MODELS

One View of Text

- Sequence of symbols (bytes, letters, characters, morphemes, words, …)
  - Let Σ denote the set of symbols.
- Lots of possible sequences. (Σ* is infinitely large.)
- Probability distributions over Σ*?

Pop Quiz

- Am I wearing a *generative* or *discriminative* hat right now?

Pop Quiz

- Generative models tell a mythical story to explain the data.
- Discriminative models focus on tasks (like sorting examples).

Trivial Distributions over Σ*

- Give probability 0 to sequences with length greater than B; uniform over the rest.
- Use data: with N examples, give probability N^{-1} to each observed sequence, 0 to the rest.

- What if we want *every* sequence to get some probability?
  - Need a probabilistic *model family* and algorithms for constructing the model from *data*.

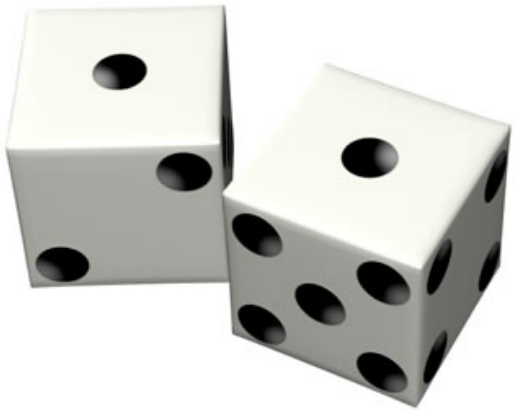
A History-Based Model

$$p(\text{start}, w_1, w_2, \ldots, w_n, \text{stop}) = \prod_{i=1}^{n+1} \gamma(w_i \mid w_1, w_2, \ldots, w_{i-1})$$

- Generate each word from left to right, conditioned on what came before it.

Die / Dice

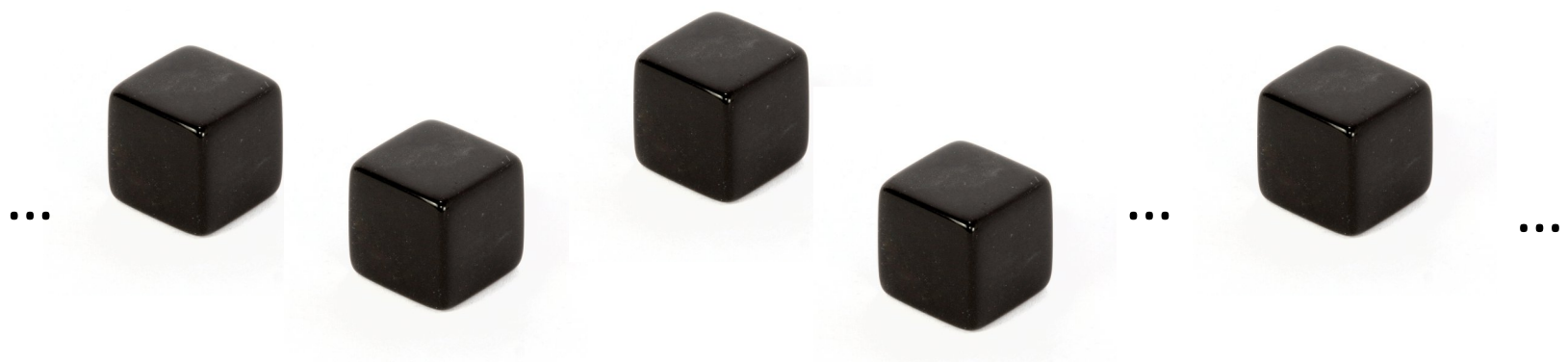
one die



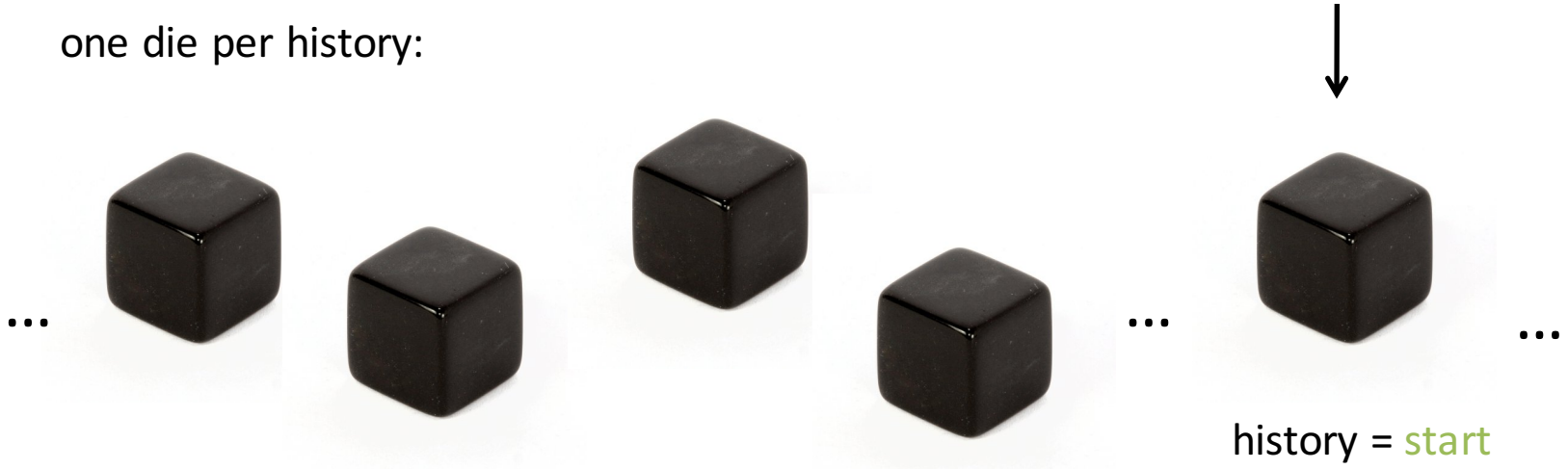
two dice

start

one die per history:

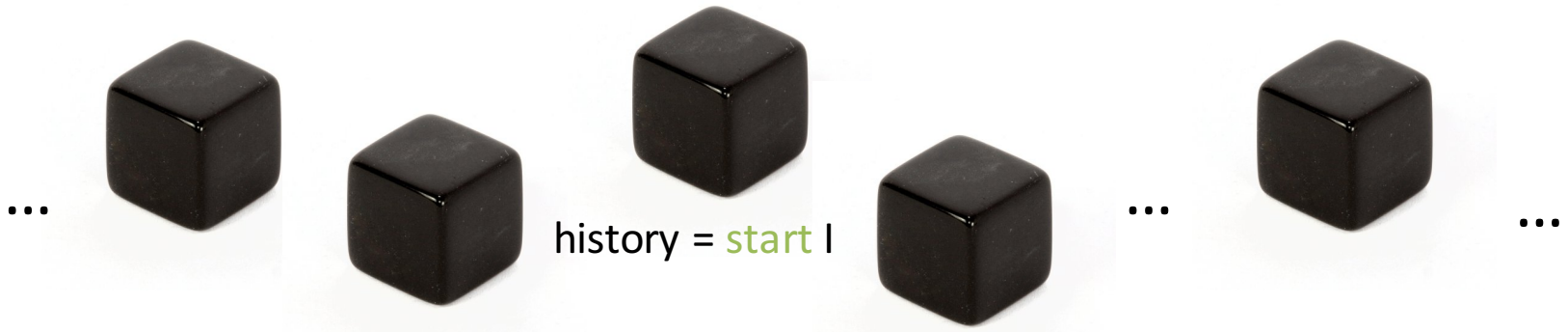


one die per history:



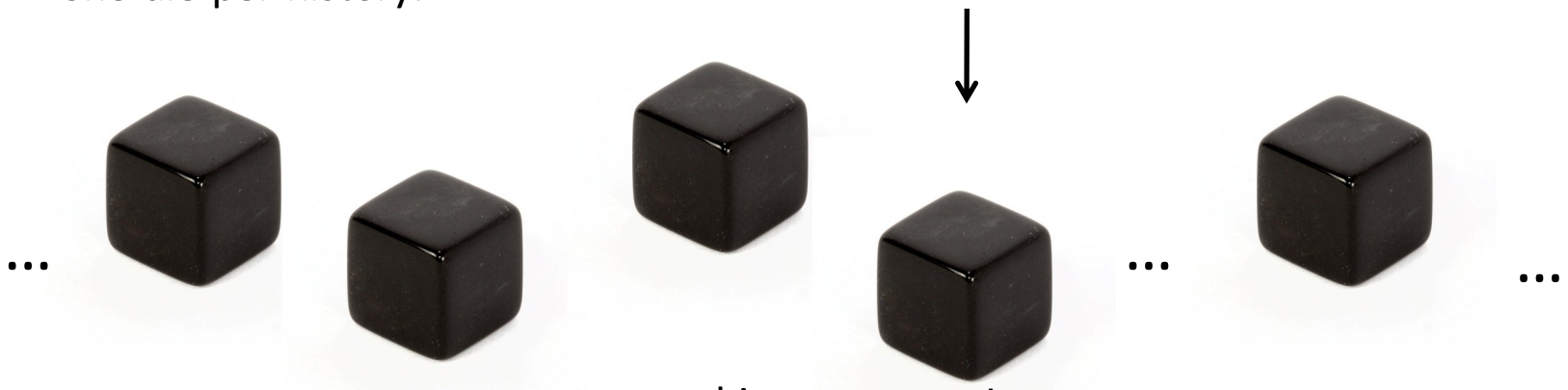
history = start

one die per history:



start I want a

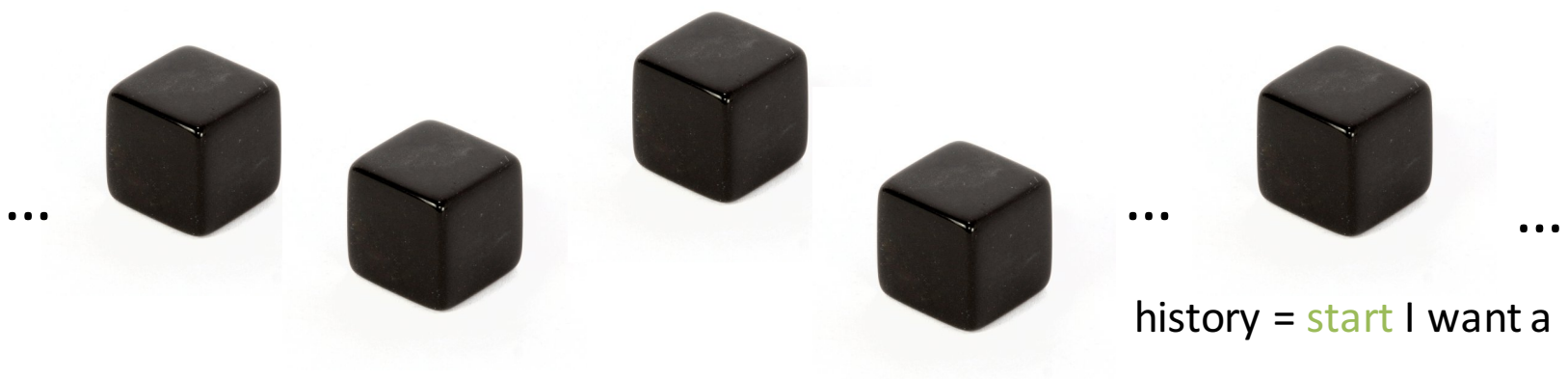
one die per history:



history = start I want

start I want a flight

one die per history:



history = start I want a

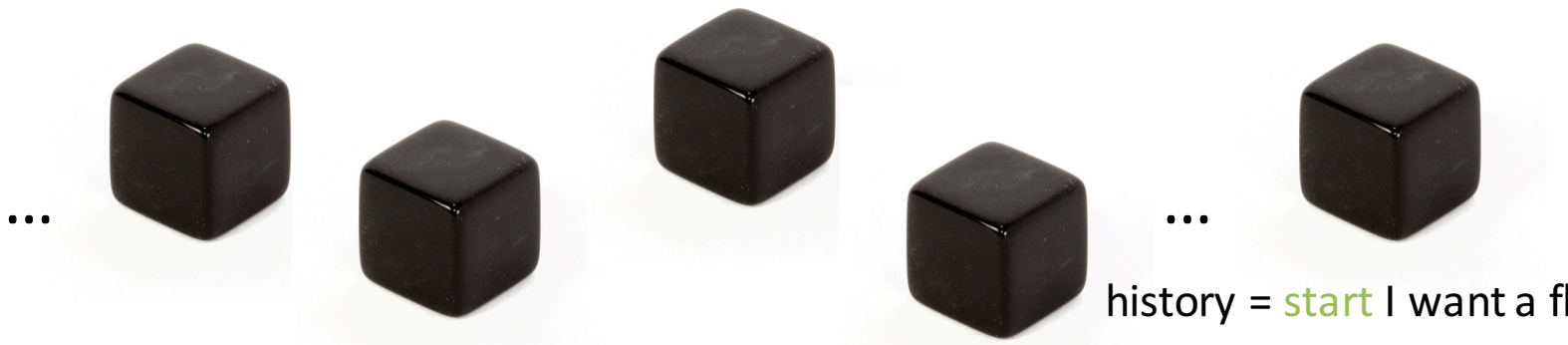
start I want a flight to

one die per history:

history = start I want a flight

start I want a flight to Lisbon

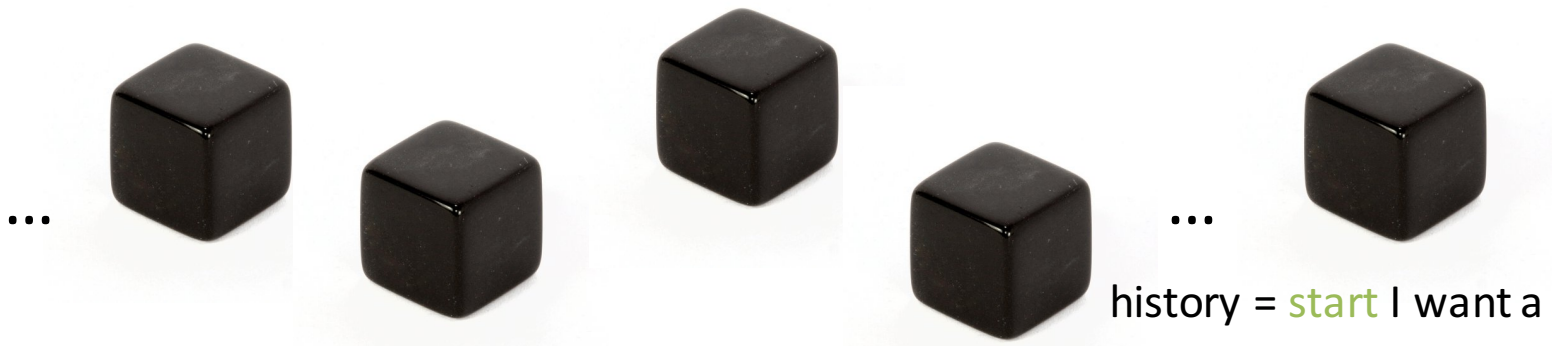
one die per history:



history = start I want a flight to

start I want a flight to Lisbon .

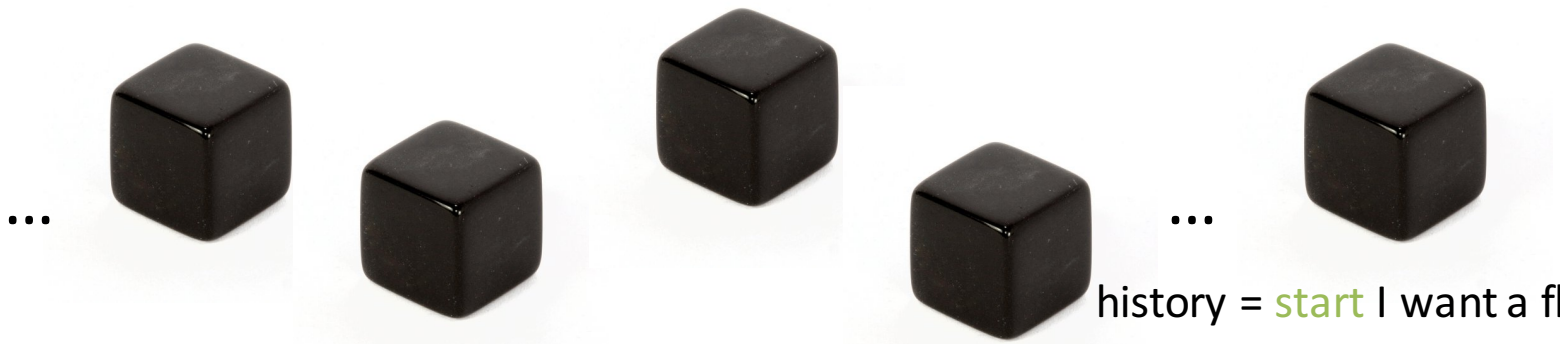
one die per history:



history = start I want a flight to Lisbon

start I want a flight to Lisbon . stop

one die per history:



history = start I want a flight to Lisbon .

A History-Based Model

$$p(\text{start}, w_1, w_2, \ldots, w_n, \text{stop}) = \prod_{i=1}^{n+1} \gamma(w_i \mid w_1, w_2, \ldots, w_{i-1})$$

- Generate each word from left to right, conditioned on what came before it.
- Very rich representational power!
- How many parameters?
- What is the probability of a sentence not seen in training data?

A Bag of Words Model

$$p(\text{start}, w_1, w_2, \ldots, w_n, \text{stop}) = \prod_{i=1}^{n+1} \gamma(w_i)$$

- Every word is independent of every other word.

start

one die:

start

one die:

start I want

one die:

start I want a

one die:

start I want a flight

one die:

start I want a flight to

one die:

start I want a flight to Lisbon

one die:

start I want a flight to Lisbon .

one die:

start I want a flight to Lisbon . stop

one die:

A Bag of Words Model

$$p(\text{start}, w_1, w_2, \ldots, w_n, \text{stop}) = \prod_{i=1}^{n+1} \gamma(w_i)$$

- Every word is independent of every other word.
- Strong assumptions mean this model cannot fit the data very closely.
- How many parameters?
- What is the probability of a sentence not seen in training data?

First Order Markov Model

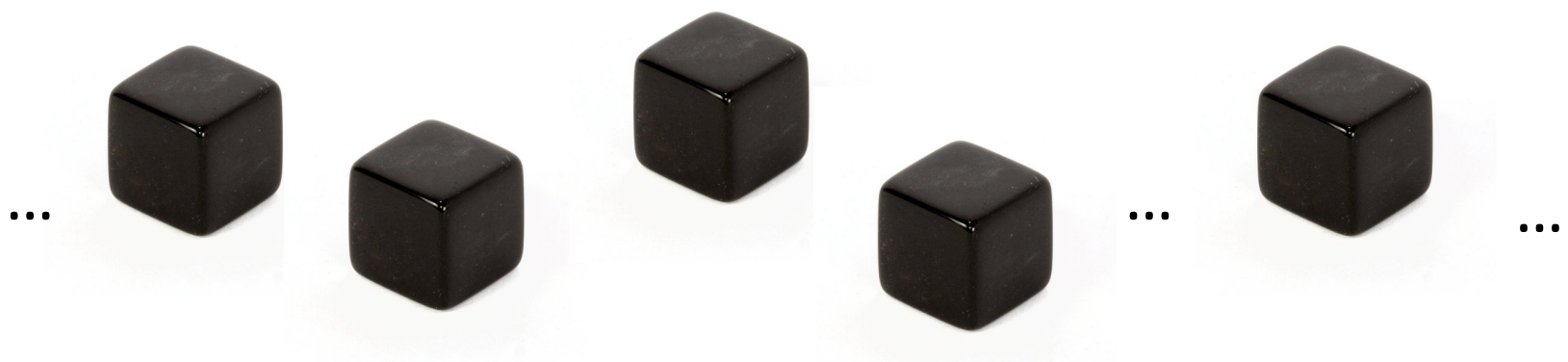
- Happy medium?

$$p(\text{start}, w_1, w_2, \ldots, w_n, \text{stop}) = \prod_{i=1}^{n+1} \gamma(w_i \mid w_{i-1})$$

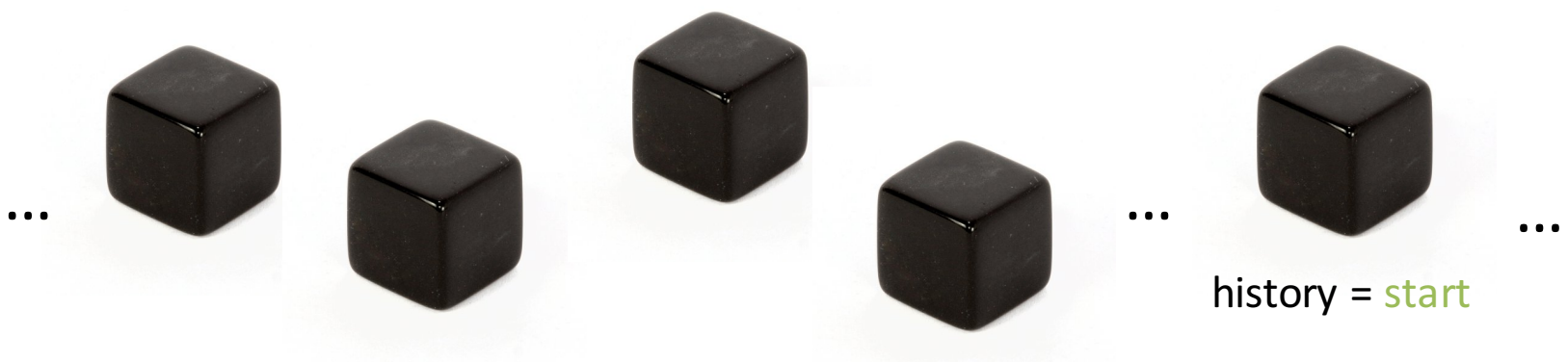
- Condition on the most recent symbol in history.

start

one die per history:

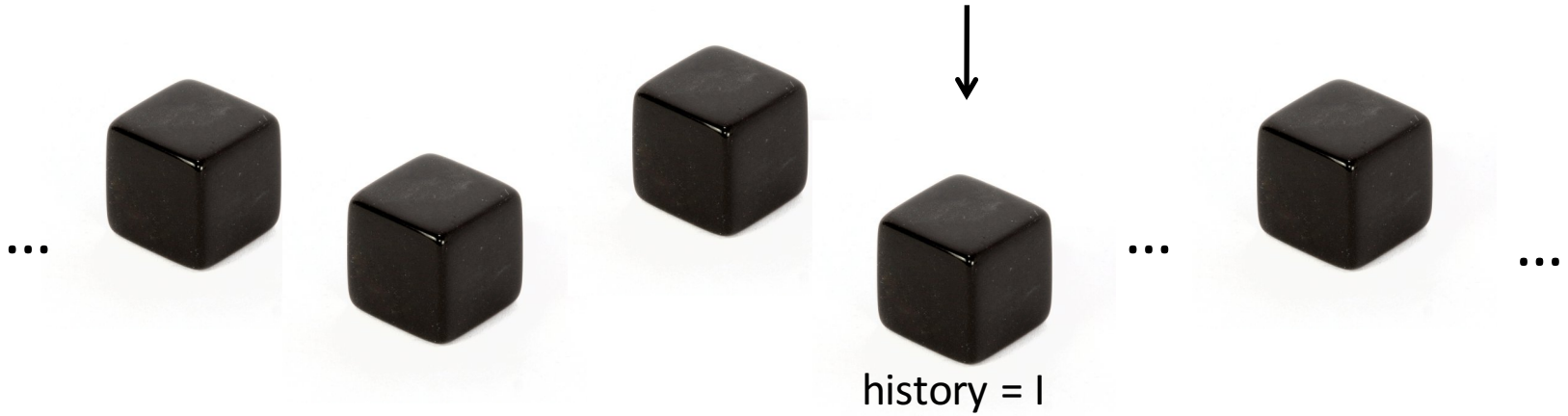


one die per history:

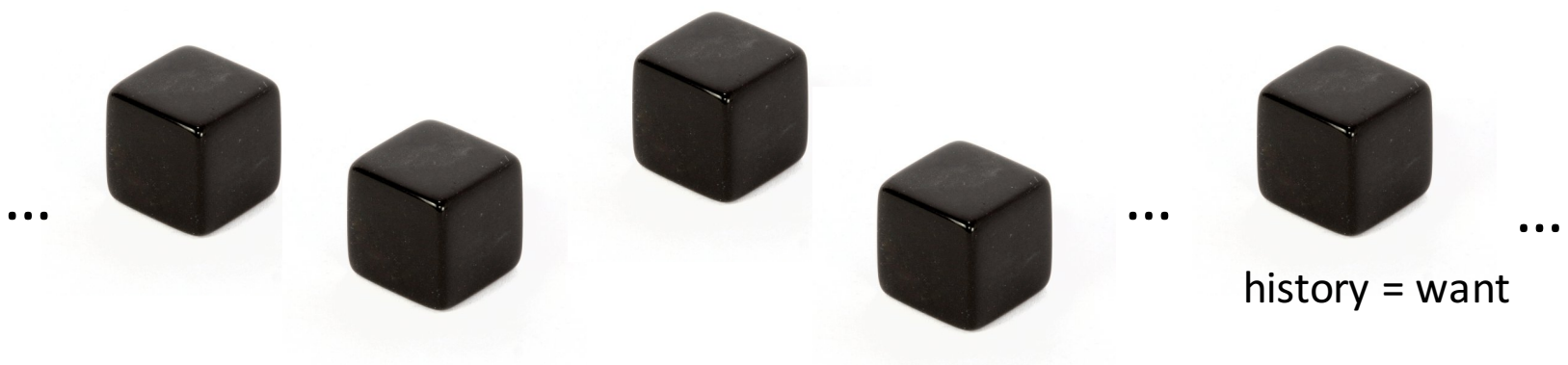


history = start

one die per history:

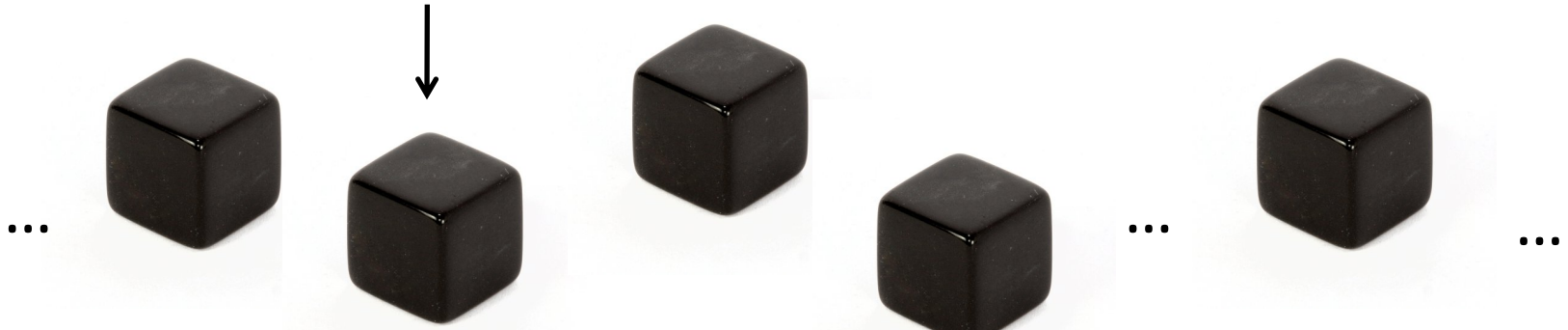


one die per history:



start I want a flight

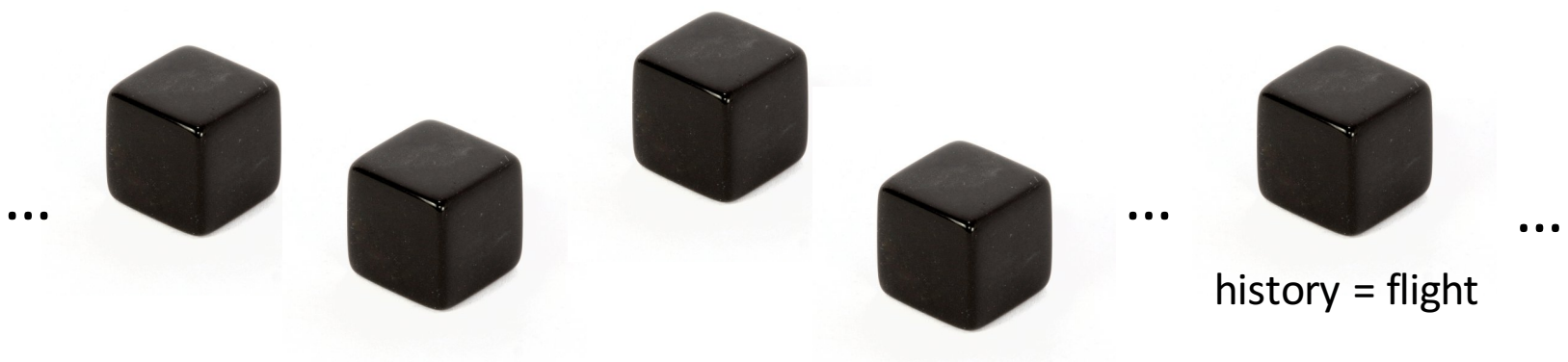
one die per history:



history = a

start I want a flight to

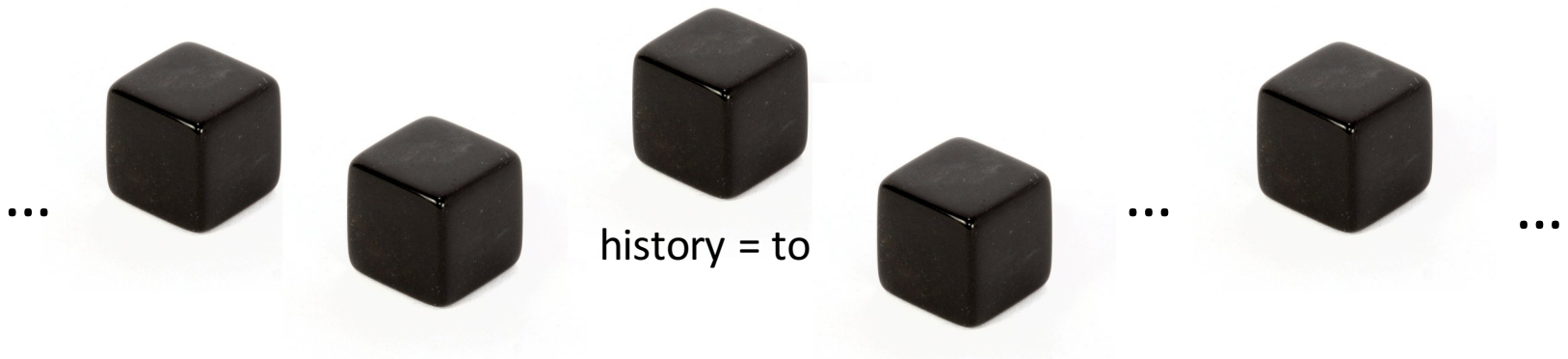
one die per history:



history = flight

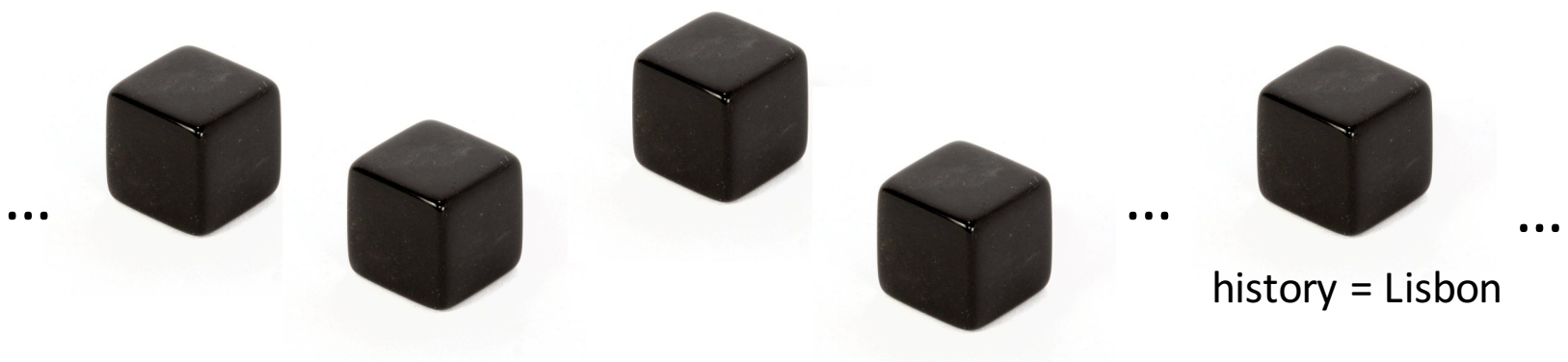
start I want a flight to Lisbon

one die per history:



start I want a flight to Lisbon .

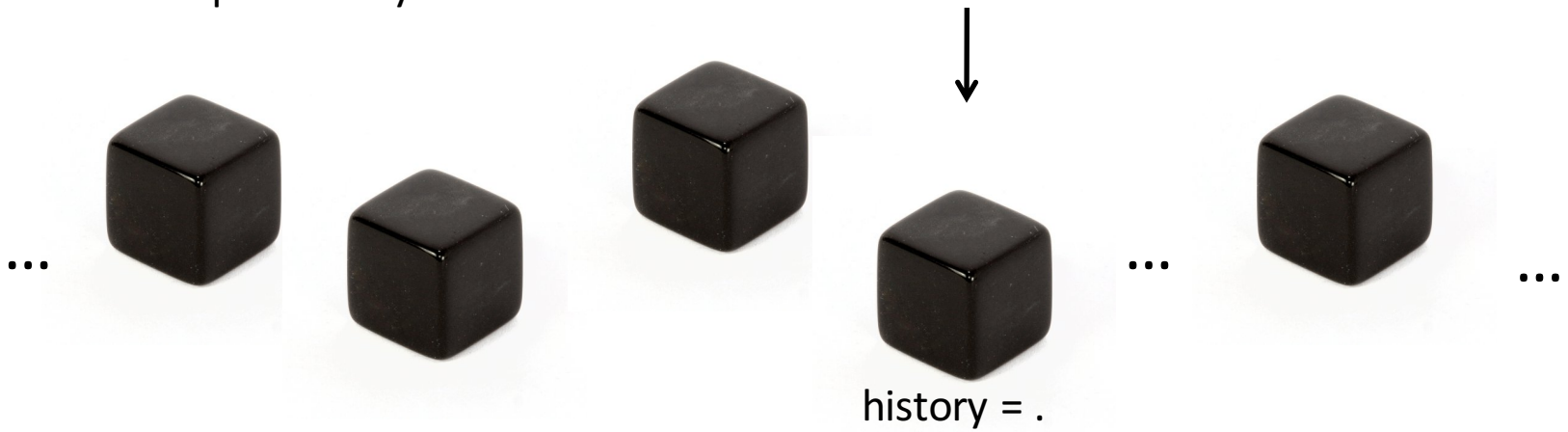
one die per history:



history = Lisbon

start I want a flight to Lisbon . stop

one die per history:



First Order Markov Model

- Happy medium?

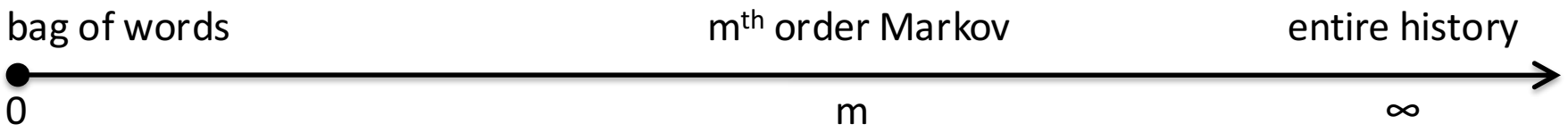
$$p(\text{start}, w_1, w_2, \ldots, w_n, \text{stop}) = \prod_{i=1}^{n+1} \gamma(w_i \mid w_{i-1})$$

- Condition on the most recent symbol in history.
- Independence assumptions?
- Number of parameters?
- Sentences not seen in training?

m^{th} Order Markov Models

$$p(\text{start}, w_1, w_2, \ldots, w_n, \text{stop}) = \prod_{i=1}^{n+1} \gamma(w_i \mid w_{i-m}, \ldots, w_{i-1})$$

(m+1)-gram 模型



fewer parameters

stronger independence assumptions

richer expressive power

Example

- Unigram model estimated on 2.8M words of American political blog text.

```
this trying our putting and funny
 and among it herring it obama
 but certainly foreign my
 c on byron again but from i
 i so and i chuck yeah the as but but republicans if
this stay oh so or it mccain bush npr this with what
and they right i while because obama
```

Example

- Bigram model estimated on 2.8M words of American political blog text.

```
the lack of the senator mccain hadn t keep this story
backwards
while showering praise of the kind of gop weakness
it was mistaken for american economist anywhere in the
white house press hounded the absence of those he s as
a wide variety of this election day after the candidate
b richardson was polled ri in hempstead moderated by
the convention that he had zero wall street journal
argues sounds like you may be the primary
but even close the bill told c e to take the obama on
the public schools and romney
fred flinstone s see how a lick skillet road it s
little sexist remarks
```

Example

- Trigram model estimated on 2.8M words of American political blog text.

```
 as i can pin them all none of them want to bet that
any of the might be
 conservatism unleashed into the privacy rule book and
when told about what paul
 fans organized another massive fundraising initiative
yesterday and i don t know what the rams supposedly
want ooh
 but she did but still victory dinner
 alone among republicans there are probably best not
all of the fundamentalist community
 asked for an independent maverick now for
crystallizing in one especially embarrassing
```

Example

- 5-gram model estimated on 2.8M words of American political blog text.

```
 he realizes fully how shallow and insincere
conservative behavior has been he realizes that there
is little way to change the situation
 this recent arianna huffington item about mccain
issuing heartfelt denials of things that were actually
true or for that matter about the shia sunni split and
which side iran was on would get confused about this
any more than someone with any knowledge of us politics
would get confused about whether neo confederates were
likely to be supporting the socialist workers party
 at the end of the world and i m not especially
discouraged now that newsweek shows obama leading by
three now
```

Example

- 100-gram model estimated on 2.8M words of American political blog text.

and it would be the work of many hands to catalogue all the ridiculous pronouncements made by this man since his long train of predictions about the middle east has been gaudily disastrously stupefyingly misinformed just the buffoon it seems for the new york times to award with a guest column for if you object to the nyt rewarding failure in quite this way then you re intolerant according to the times editorial page editor andrew rosenthal rosenthal doesn t seem to recognize that his choice of adjectives to describe kristol serious respected are in fact precisely what is at issue for those whom he dismisses as having a fear of opposing views

(m+1)-Gram Models

Pros

- Easily understood **linguistic formalism**.
- Fully generative.
- Algorithms:
  - calculate probability of a sequence
  - choose a sequence from a set
  - training

Cons

- Obviously inaccurate linguistic formalism.
- As m grows, data sparseness becomes a problem.
  - Smoothing is a black art.
- How to deal with unknown words?

(m+1)-Gram Models

Pros

- Easily understood **linguistic formalism**.
- Fully generative.
- Algorithms:
  - calculate probability of a sequence
  - choose a sequence from a set
  - training

Cons

- Obviously inaccurate linguistic formalism.
- As m grows, data sparseness becomes a problem.
  - Smoothing is a black art.
- How to deal with unknown words?

Calculating the Probability of a Sequence

- Let n be the length of the sequence and m be the length of the history.
- For every consecutive (m+1) words $w_i \ldots w_{i+m}$, look up $p(w_{i+m} \mid w_i \ldots w_{i+m-1})$.
- Look up $p(\text{stop} \mid w_{n-m} \ldots w_n)$.
- Multiply these quantities together.

Choosing a Sequence from a Set

- Calculate the probability of each sequence in the set.
- Choose the one that has the highest probability.

Training

- Maximum likelihood estimation by relative frequencies:

unigram $$\hat{\gamma}(w) = \frac{\mathrm{freq}(w)}{\#\mathrm{words}}$$

bigram $$\hat{\gamma}(w \mid w') = \frac{\mathrm{freq}(w'w)}{\mathrm{freq}(w')}$$

trigram $$\hat{\gamma}(w \mid w'w'') = \frac{\mathrm{freq}(w'w''w)}{\mathrm{freq}(w'w'')}$$

general $$\hat{\gamma}(w \mid \boldsymbol{h}) = \frac{\mathrm{freq}(\boldsymbol{h}w)}{\mathrm{freq}(\boldsymbol{h})}$$

Note about Recent Research

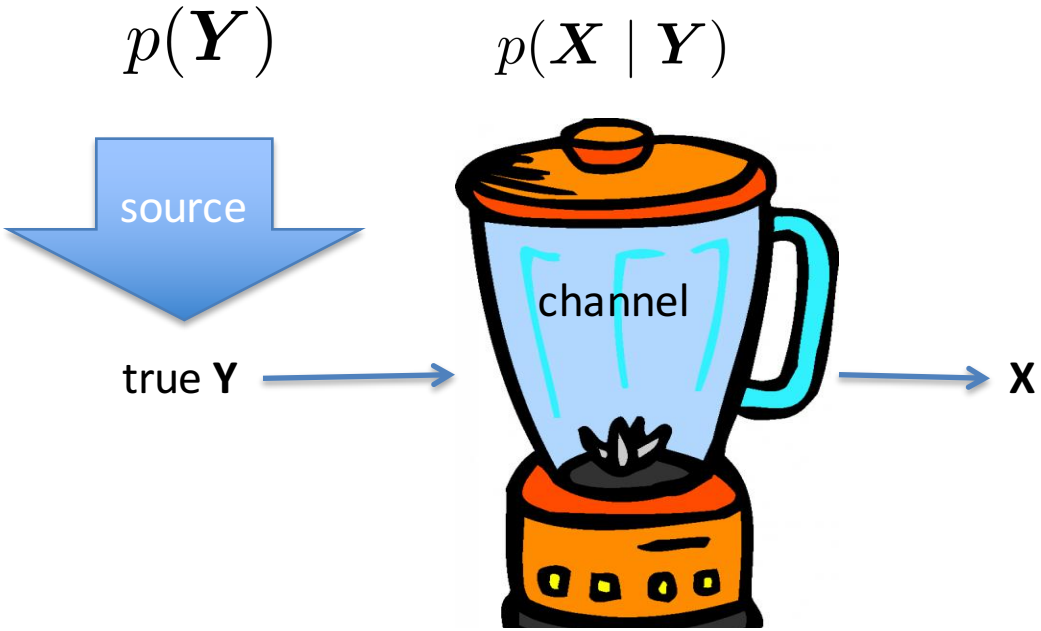
- "Web-scale" n-gram models
  - More data always seem to make language models better (Brants et al., 2007, *inter alia*)
- Efficient implementations
  - Runtime: MapReduce architectures (e.g., Lin & Dyer, 2010)
  - Memory: compression (e.g., Heafield, 2011)
- Neural language models (e.g., Mikolov et al., 2010)
  - Use continuous word representations and recurrent functions (e.g., "LSTM") to define the probability of the next word.
  - n need not be fixed!

Sequence Models as Components

- Typically we care about a sequence together with something else.
  - Analysis: sequence in, predict "something else."
  - Generation: "something else in," sequence out.

- Sequence models are useful components in *both* scenarios.

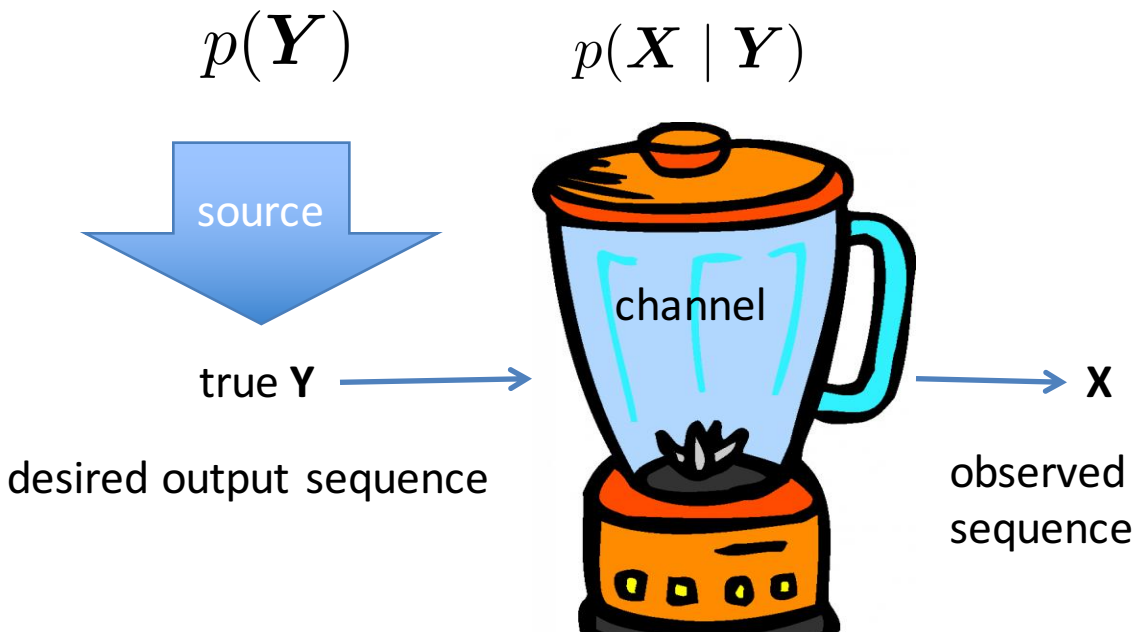
Noisy Channel



decoding rule:

$$\hat{\boldsymbol{y}} = \arg\max_{\boldsymbol{y}} p(\boldsymbol{y} \mid \boldsymbol{x}) = \arg\max_{\boldsymbol{y}} p(\boldsymbol{x} \mid \boldsymbol{y}) \times p(\boldsymbol{y})$$

Sequence Model as Source

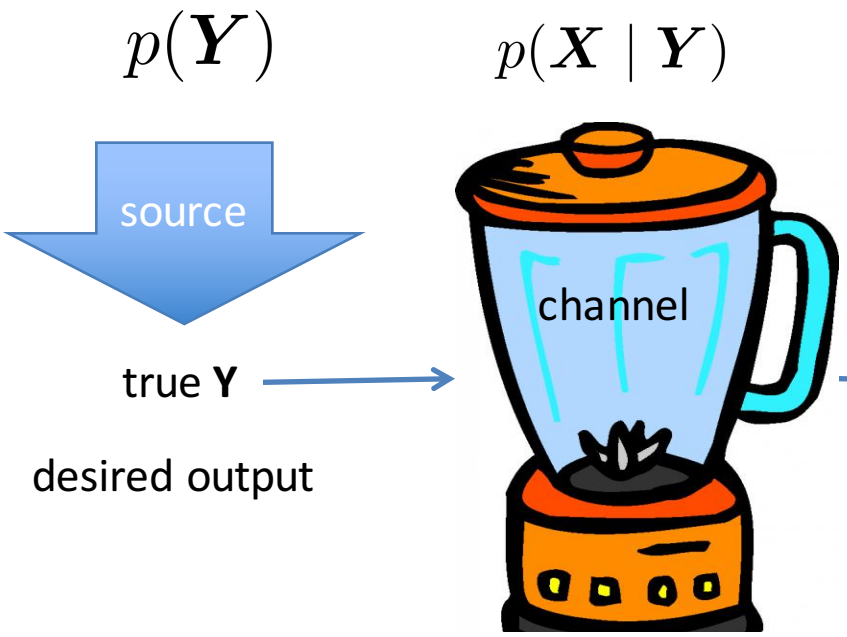


- speech recognition (Jelinek, 1997)
- machine translation (Brown et al., 1993)
- optical character recognition (Kolak and Resnik, 2002)
- spelling and punctuation correction (Kernighan et al., 1990)

decoding rule:

$$\hat{\boldsymbol{y}} = \arg\max_{\boldsymbol{y}} p(\boldsymbol{y} \mid \boldsymbol{x}) = \arg\max_{\boldsymbol{y}} p(\boldsymbol{x} \mid \boldsymbol{y}) \times p(\boldsymbol{y})$$

Sequence Model as Channel



- text categorization
- language identification
- information retrieval (Ponte and Croft, 1998; Berger and Lafferty, 1999)
- sentence compression (Knight and Marcu, 2002)
- question to search query (Radev et al., 2001)

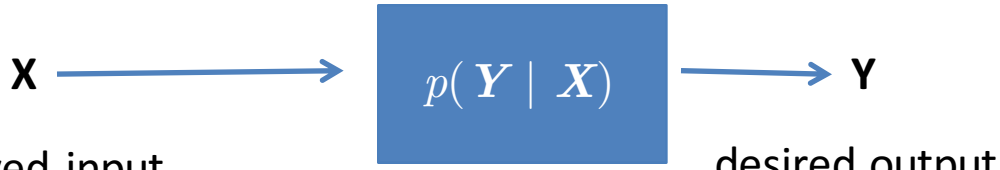
decoding rule:

$$\hat{\boldsymbol{y}} = \arg\max_{\boldsymbol{y}} p(\boldsymbol{y} \mid \boldsymbol{x}) = \arg\max_{\boldsymbol{y}} p(\boldsymbol{x} \mid \boldsymbol{y}) \times p(\boldsymbol{y})$$

Sequence Model as a Direct Model

Recent models of …

- dialog
- machine translation



observed input
(not modeled!)

desired output

decoding rule:

$$\hat{\boldsymbol{y}} = \arg\max_{\boldsymbol{y}} p(\boldsymbol{y} \mid \boldsymbol{x})$$

It's Hard to Beat N-Grams!

- They are very fast to work with. They fit the data really, really well.
- Improvements for *some specific problems* follow from:
  - task-specific knowledge
  - domain knowledge (e.g., linguistics)

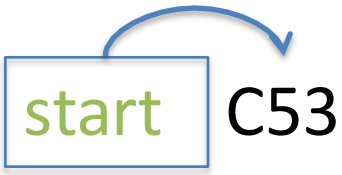
Class-Based Sequence Models

- From Brown et al. (1990):

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

- "cl" is a deterministic function from words to a smaller set of classes.
  - Each word only gets one class; known in advance.
  - Discovered from data using a clustering algorithm.

start



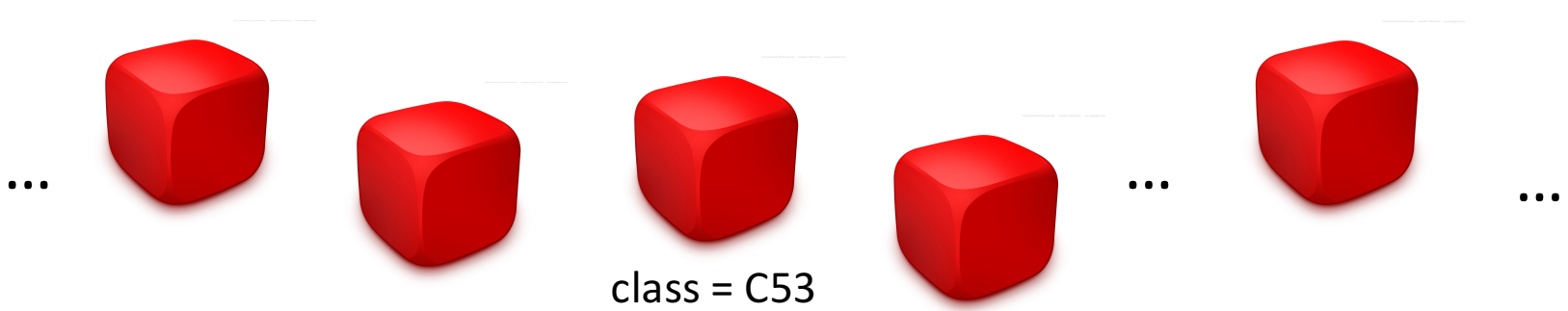
one “next class” die per class:

Each word appears on only one of the word dice.

start C53

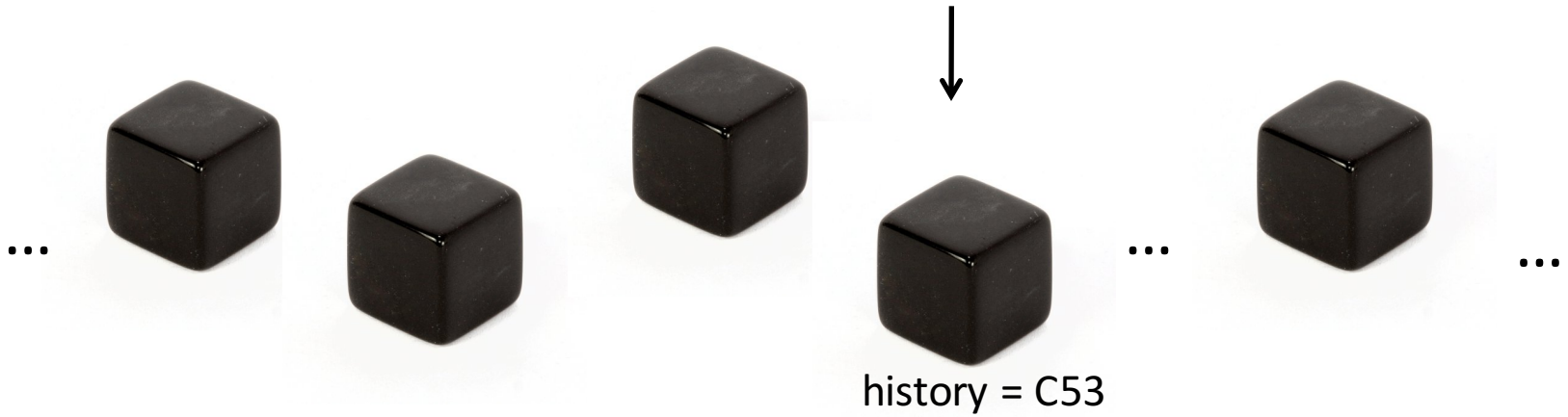
I

one word die per class:



I

one “next class” die per class:

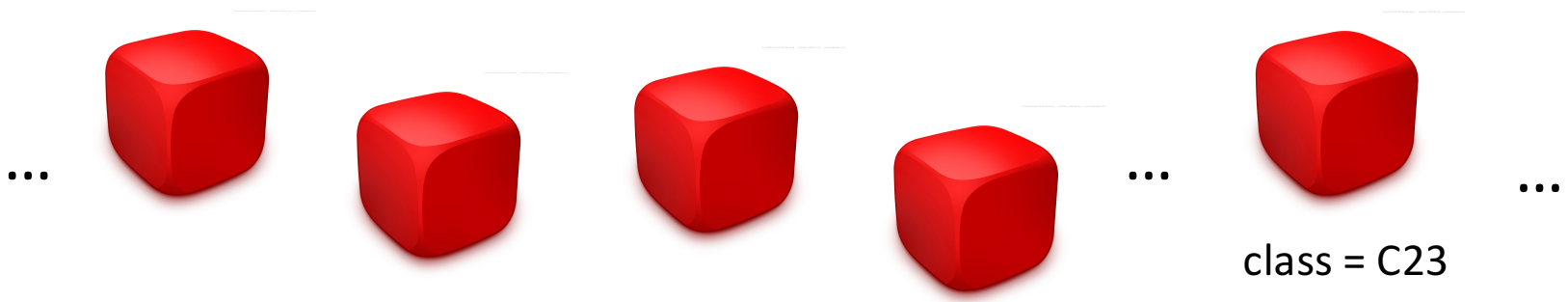


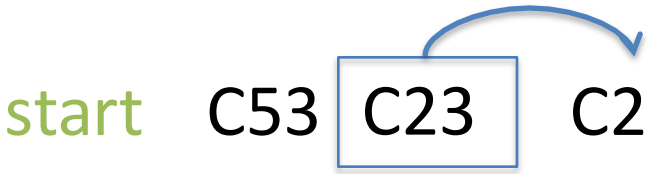
history = C53

start C53 C23

I want

one word die per class:





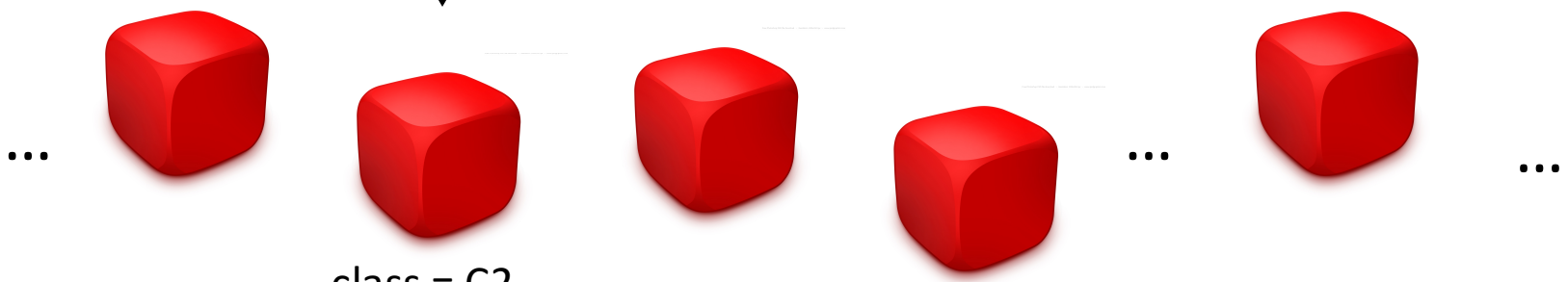
I want

one “next class” die per class:

start C53 C23 C2

I want a

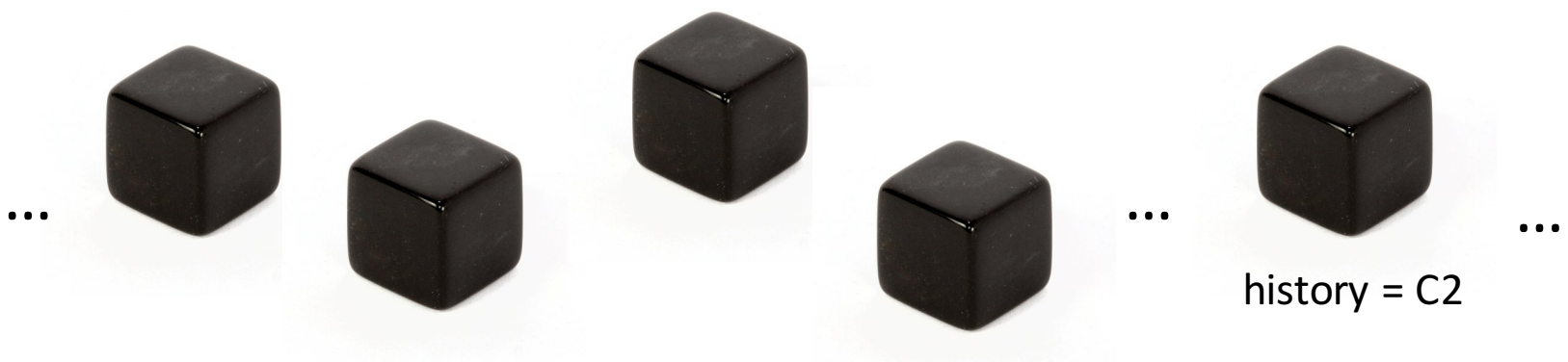
one word die per class:



start C53 C23 C2 C5

I want a

one “next class” die per class:

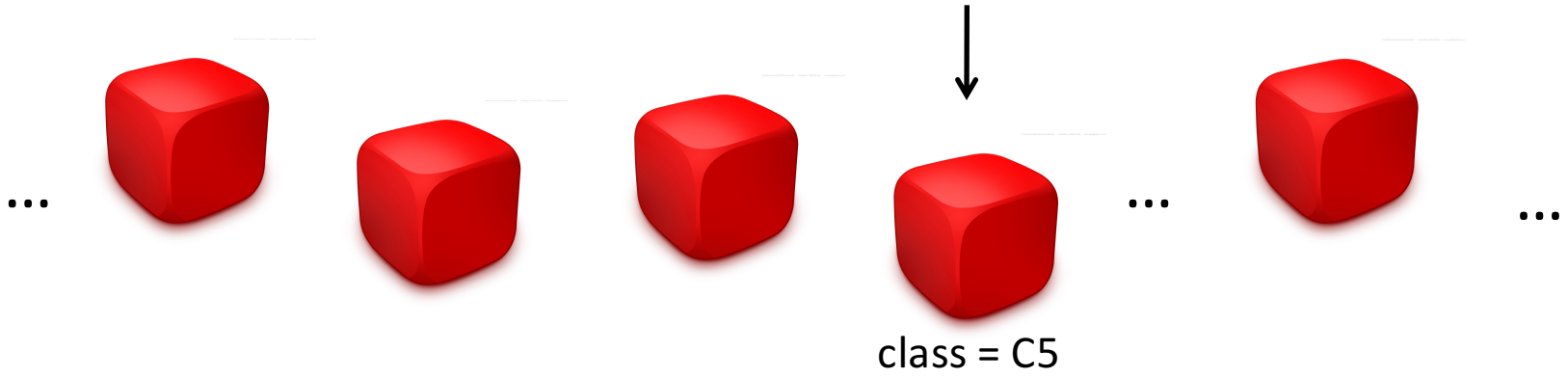


history = C2

start C53 C23 C2 C5

I want a flight

one word die per class:



Class-Based Sequence Models

- From Brown et al. (1990):

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

- Independence assumptions?
- Number of parameters?
- Generalization ability?

Lecture Outline

- ✓ Markov models
2. Hidden Markov models
3. Viterbi algorithm
4. Other inference algorithms for HMMs
5. Learning algorithms for HMMs

HIDDEN MARKOV MODELS

Hidden Markov Model

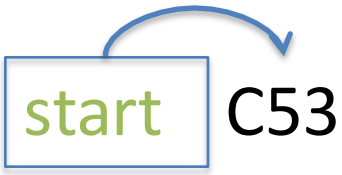
- A model over sequences of symbols, but there is missing information associated with each symbol: its "state."
  - Assume a finite set of possible states, Λ.

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

emission transition

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

eta



C53

one “next class” die per class:

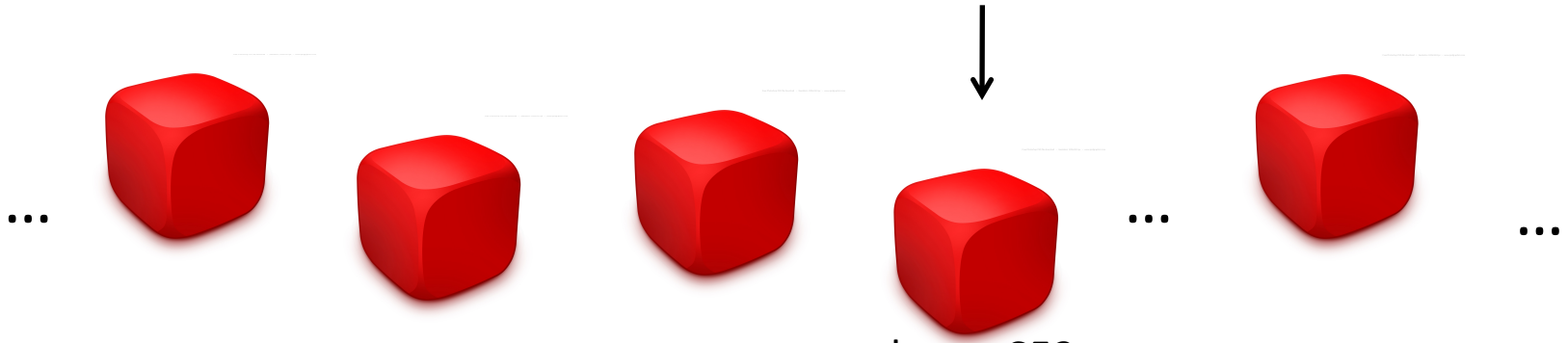
history = start

The only change to the class-based model is that now, the different dice can *share words*!

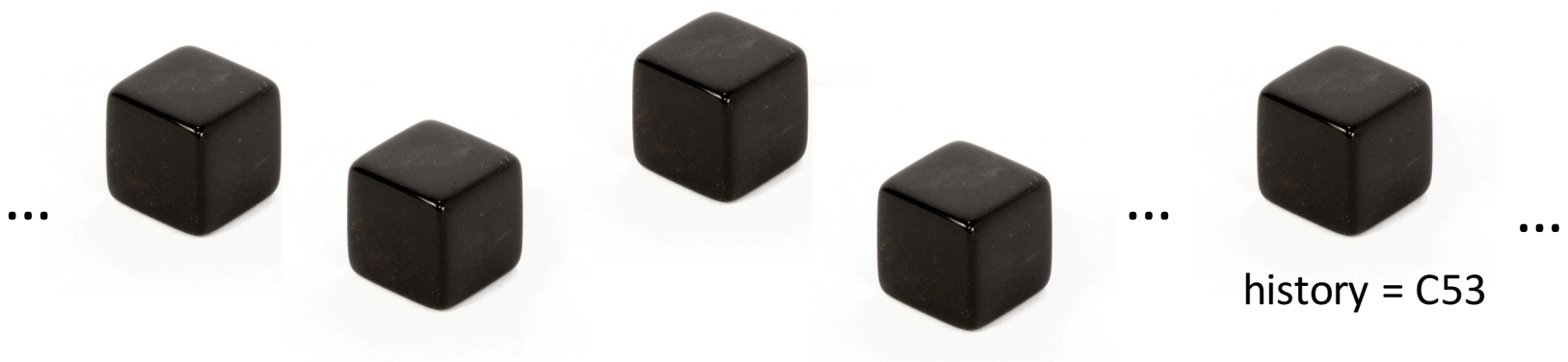
start C53

I

one word die per class:



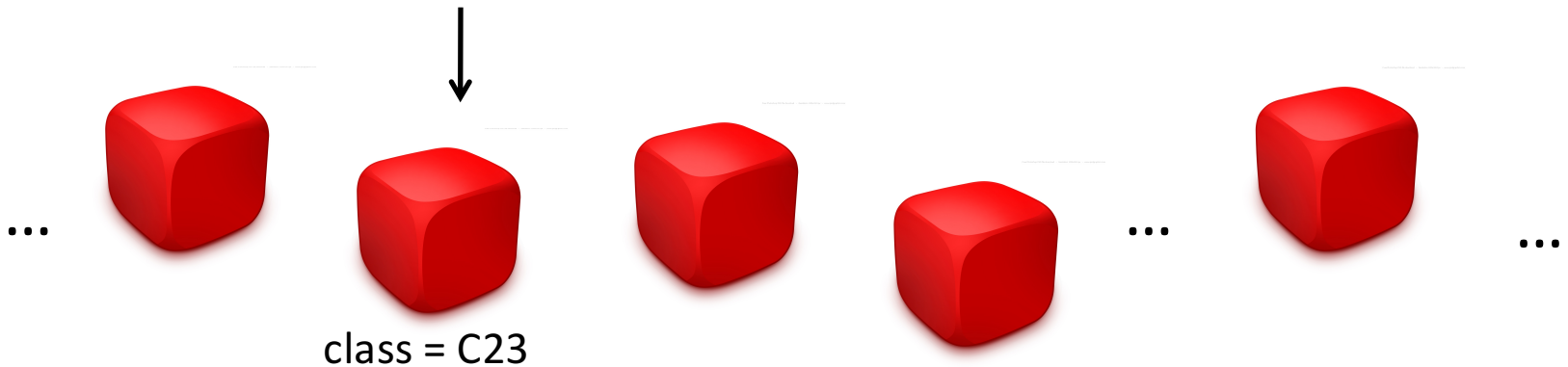
one “next class” die per class:

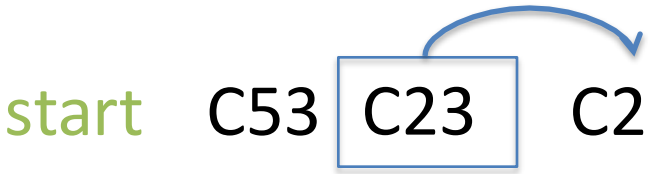


start C53 C23

I want

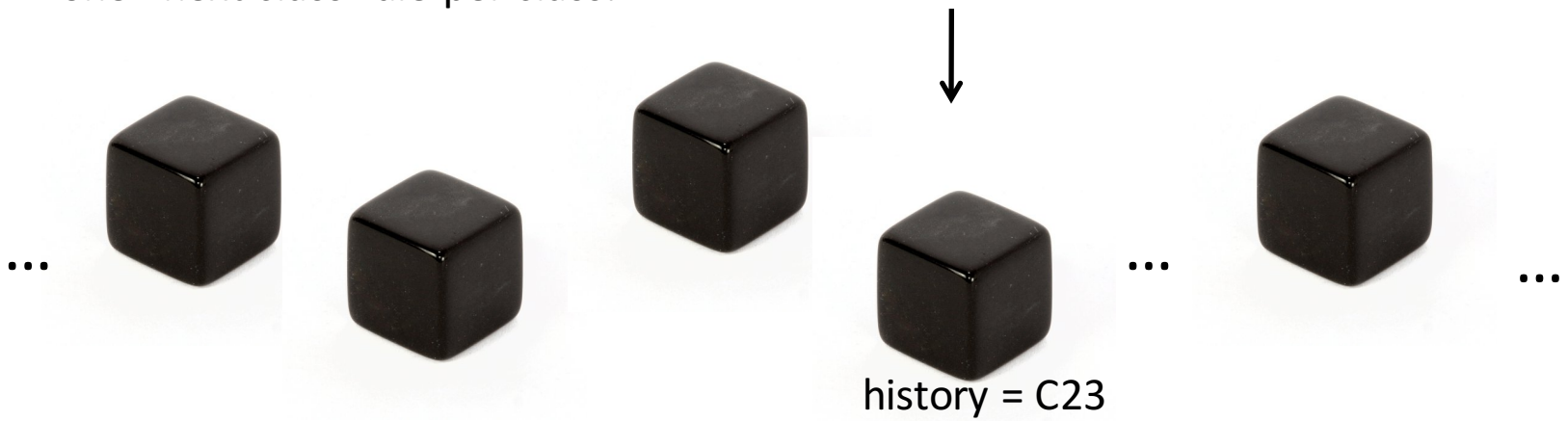
one word die per class:





I want

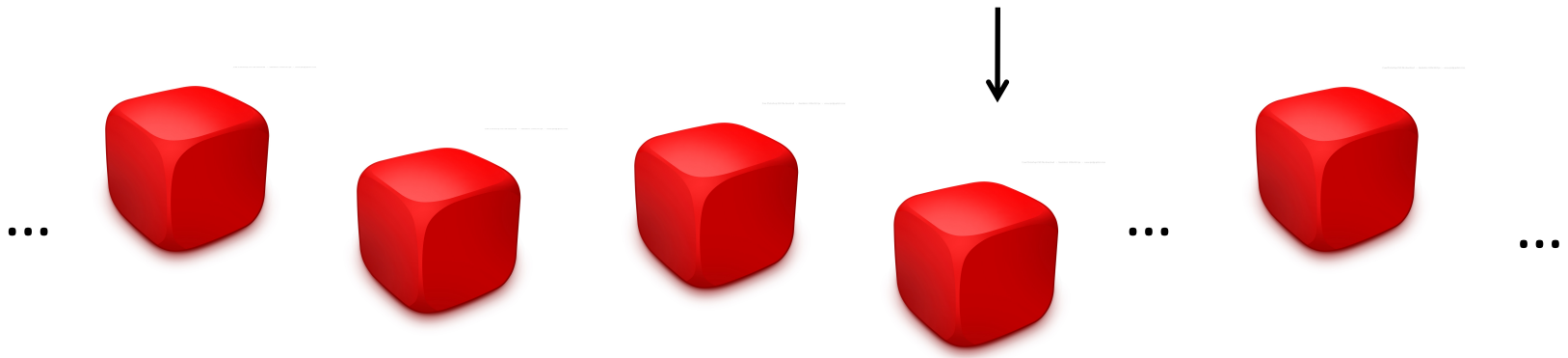
one “next class” die per class:



start C53 C23 C2

I want a

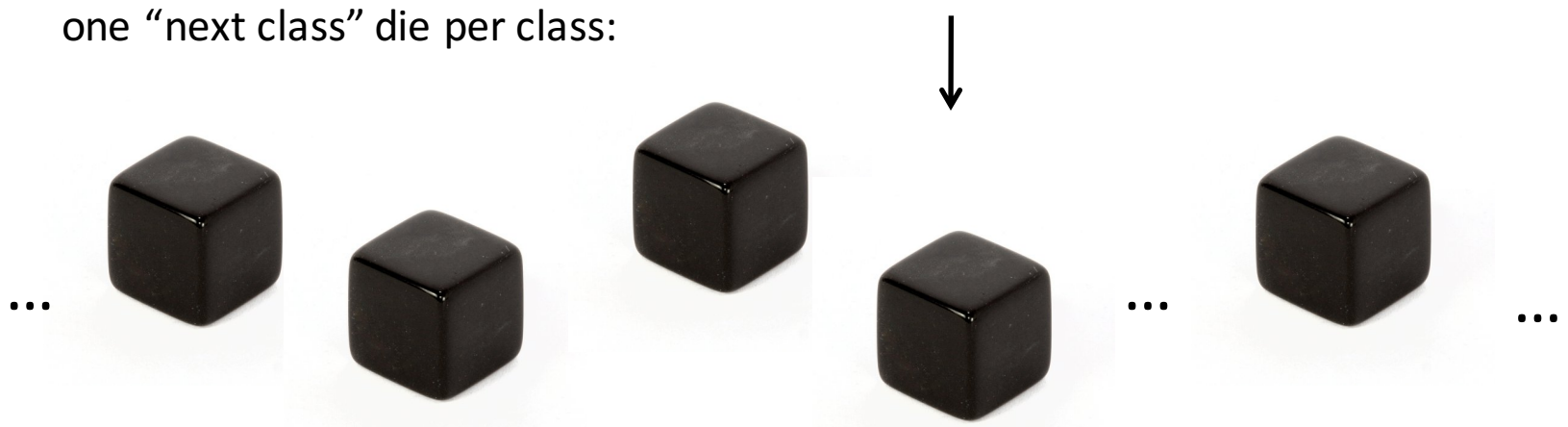
one word die per class:



start C53 C23 C2 C5

I want a

one “next class” die per class:

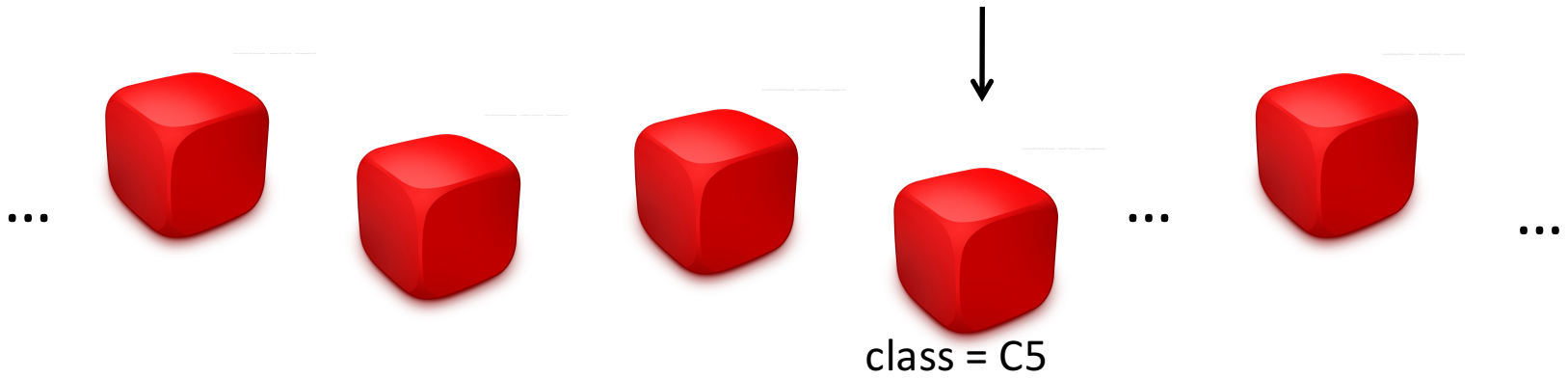


history = C2

start C53 C23 C2 C5

I want a flight

one word die per class:



Two Equivalent Stories

- First, as shown: transition, emit, transition, emit, transition, emit.

- Second:
  - Generate the sequence of transitions. Essentially, a Markov model on classes.
  - Stochastically replace each class with a word.

mth Order Hidden Markov Models

- We can condition on a longer history of past states:

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

- Number of parameters?
- Benefit: longer "memory."
- Today I will stick with first-order HMMs.

Uses of HMMs in NLP

- Part-of-speech tagging (Church, 1988; Brants, 2000)
- Named entity recognition (Bikel et al., 1999) and other information extraction tasks
- Text chunking and shallow parsing (Ramshaw and Marcus, 1995)
- Word alignment in parallel text (Vogel et al., 1996)

- Also popular in computational biology and central to speech recognition.

Part of Speech Tagging

After paying the medical bills , Frances was nearly broke .

RB VBG DT JJ NNS , NNP VBZ RB JJ .

- Adverb (RB)
- Verb (VBG, VBZ, and others)
- Determiner (DT)
- Adjective (JJ)
- Noun (NN, NNS, NNP, and others)
- Punctuation (., ,, and others)

Named Entity Recognition

With Commander Chris Ferguson at the helm ,

Atlantis touched down at Kennedy Space Center .

Named Entity Recognition

O B-person I-person I-person O O O O

With Commander Chris Ferguson at the helm ,

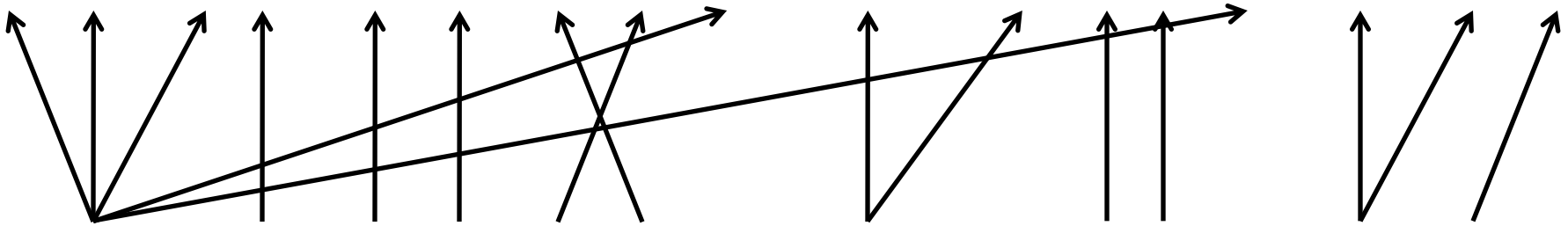
B-space-shuttle O O O B-place I-place I-place O

Atlantis touched down at Kennedy Space Center .

- What makes this hard?

Word Alignment

Mr. President , Noah’s ark was filled not with production factors , but with living creatures.



NULL Noahs Arche war nicht voller Productionsfactoren , sondern Geschöpfe .

Word Alignment

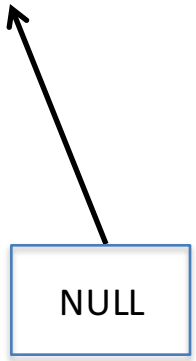
Mr. President , Noah’s ark was filled not with production factors , but with living creatures.

NULL

Noahs Arche war nicht voller Productionsfactoren , sondern Geschöpfe .

Word Alignment

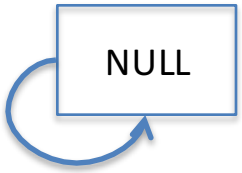
Mr. President , Noah’s ark was filled not with production factors , but with living creatures.



Noahs Arche war nicht voller Productionsfactoren , sondern Geschöpfe .

Word Alignment

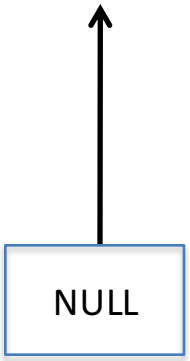
Mr. President , Noah’s ark was filled not with production factors , but with living creatures.



Noahs Arche war nicht voller Productionsfactoren , sondern Geschöpfe .

Word Alignment

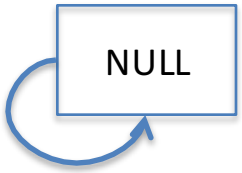
Mr. President , Noah’s ark was filled not with production factors , but with living creatures.



Noahs Arche war nicht voller Productionsfactoren , sondern Geschöpfe .

Word Alignment

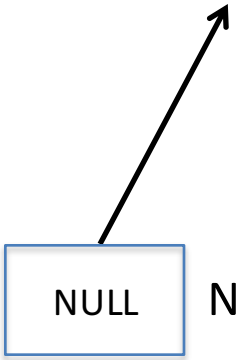
Mr. President , Noah’s ark was filled not with production factors , but with living creatures.



Noahs Arche war nicht voller Productionsfactoren , sondern Geschöpfe .

Word Alignment

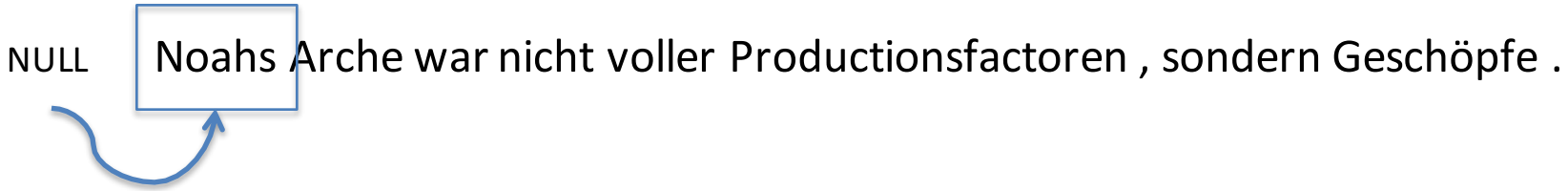
Mr. President , Noah’s ark was filled not with production factors , but with living creatures.



Noahs Arche war nicht voller Productionsfactoren , sondern Geschöpfe .

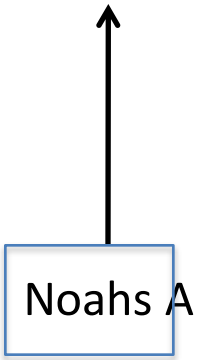
Word Alignment

Mr. President , Noah’s ark was filled not with production factors , but with living creatures.



Word Alignment

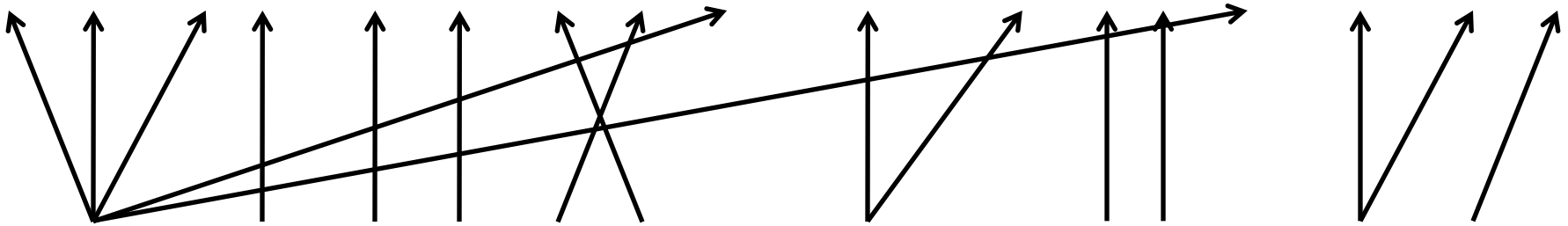
Mr. President , Noah’s ark was filled not with production factors , but with living creatures.



NULL Noahs Arche war nicht voller Productionsfactoren , sondern Geschöpfe .

Word Alignment

Mr. President , Noah’s ark was filled not with production factors , but with living creatures.



NULL Noahs Arche war nicht voller Productionsfactoren , sondern Geschöpfe .

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.

Lecture Outline

- ✓ Markov models
- ✓ Hidden Markov models
3. Viterbi algorithm
4. Other inference algorithms for HMMs
5. Learning algorithms for HMMs

ALGORITHMS FOR HIDDEN MARKOV MODELS

How To Calculate …

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 sequences:

5. The parameters of the HMM (**γ** and **η**)?

Problem 1:
Most Likely State Sequence

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

- 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

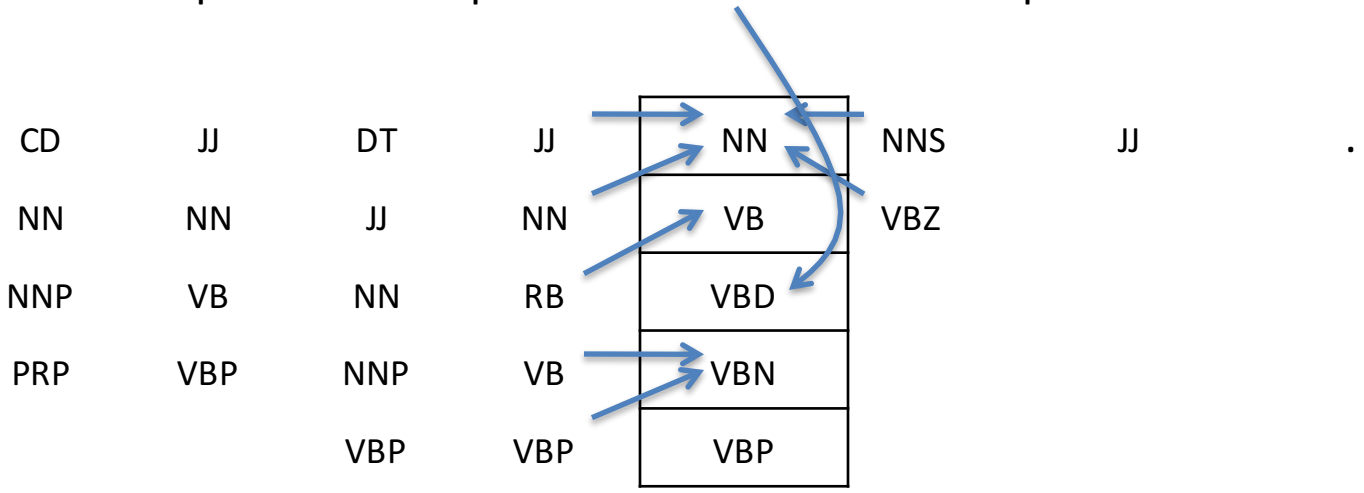
- List all the possibilities in Λ^n.
  - Correct.
  - Inefficient.
- Work left to right and greedily pick the best s_i at each point, based on s_{i-1} and w_i.
  - Not correct; solution may not be equal to:

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

  - But fast!

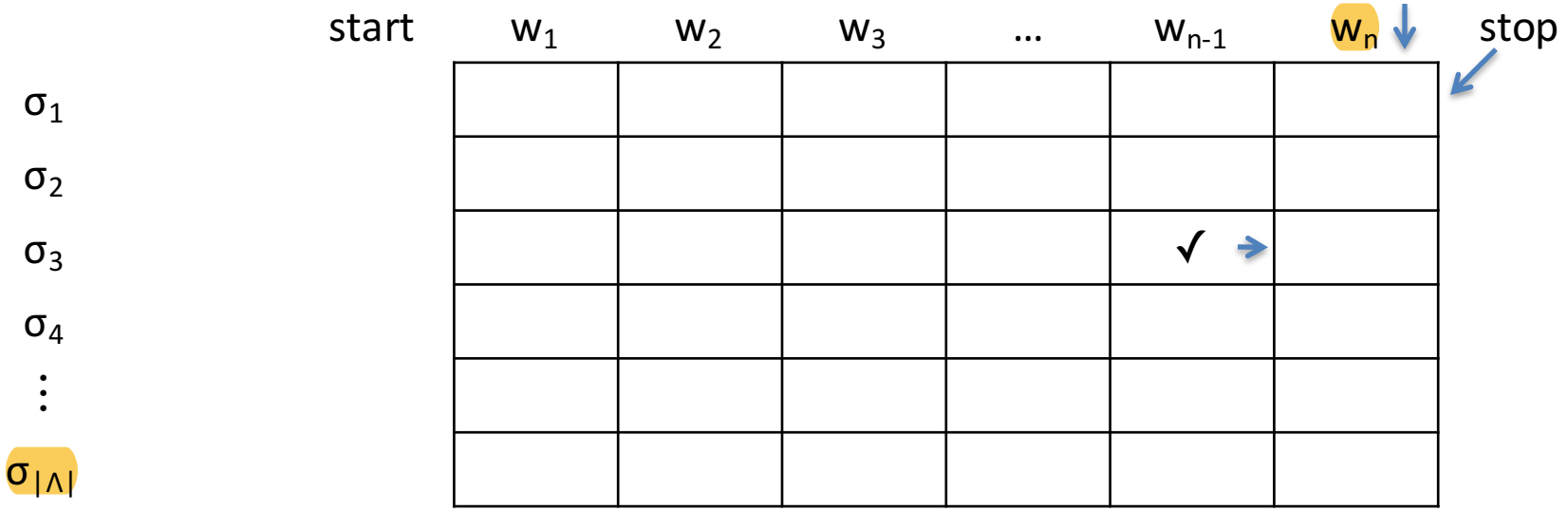
Interactions

- 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



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

概率

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 one 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.
  - 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.

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

注意 \delta 是一列，有很多个

Viterbi Algorithm (Scores Only)

- For every σ in Λ, let:

\delta 是一列，有很多个

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

- 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

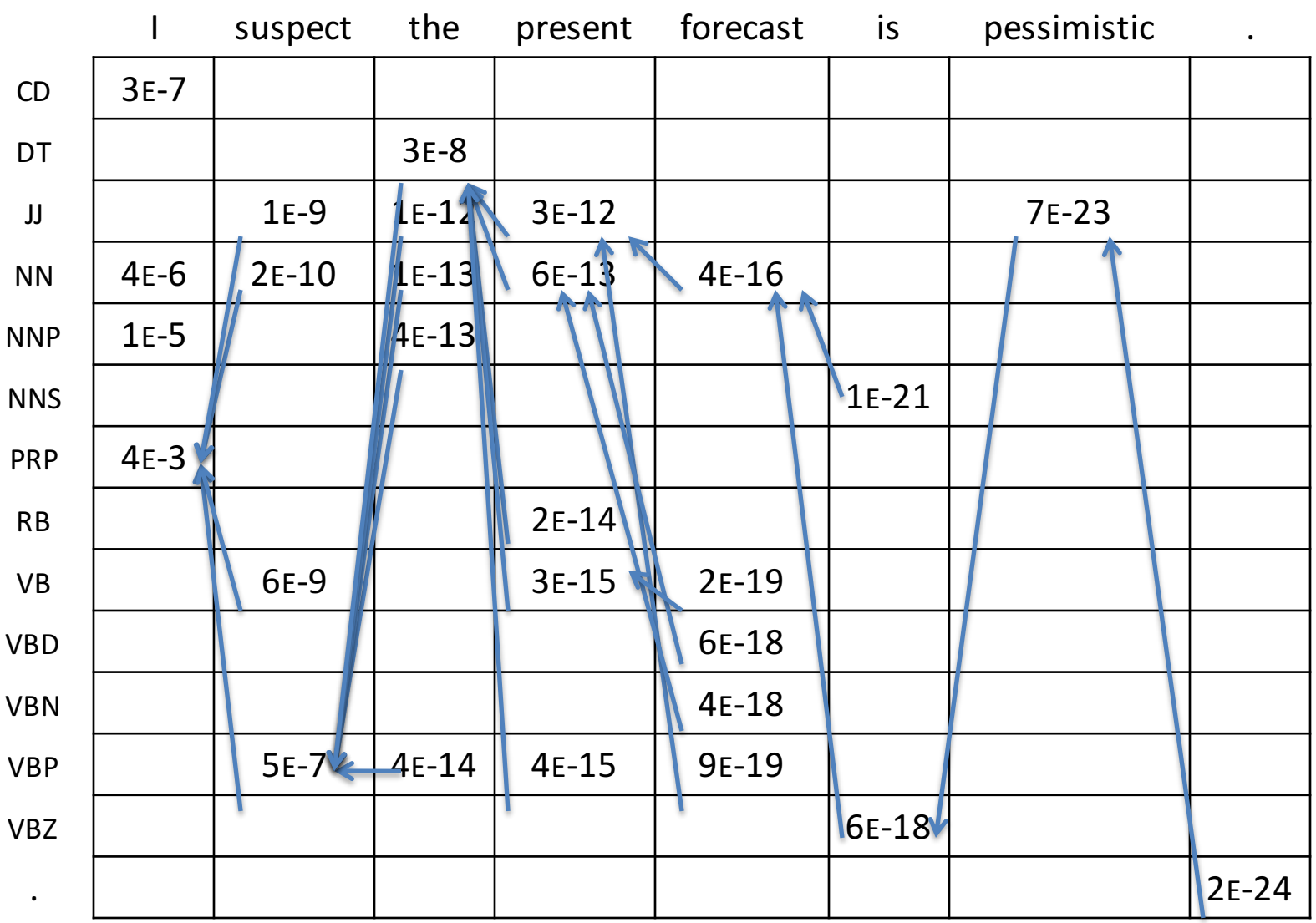
- As described, this algorithm only lets us calculate the *probability* of the best label sequence.
- It does not recover the best sequence!

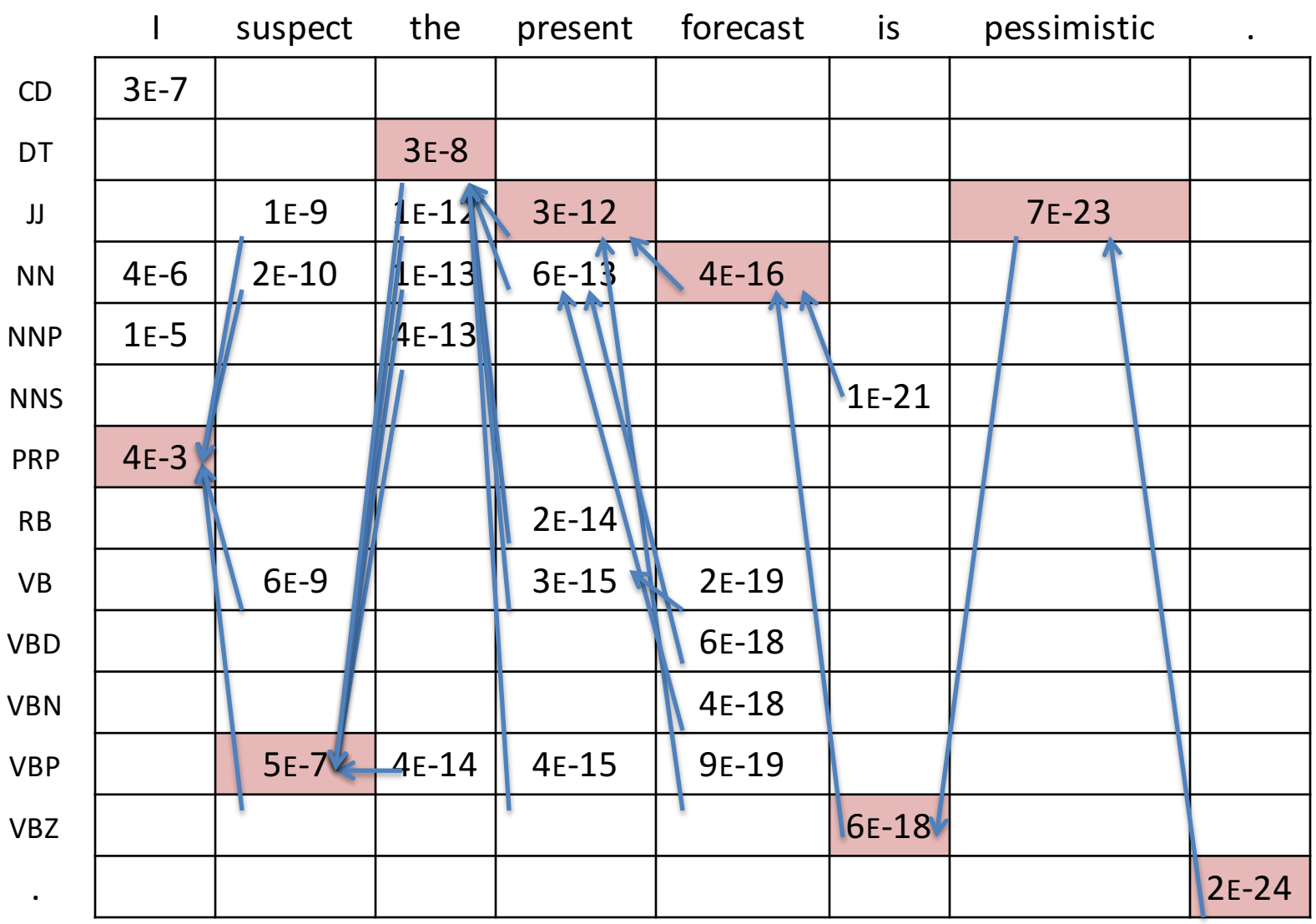
Understanding the Scores

- $\text{score}_i(\sigma)$ 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: Scores & Backpointers

- 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: 记住这个最大值从前面哪个delta来

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

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

Viterbi Algorithm: Backtrace

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

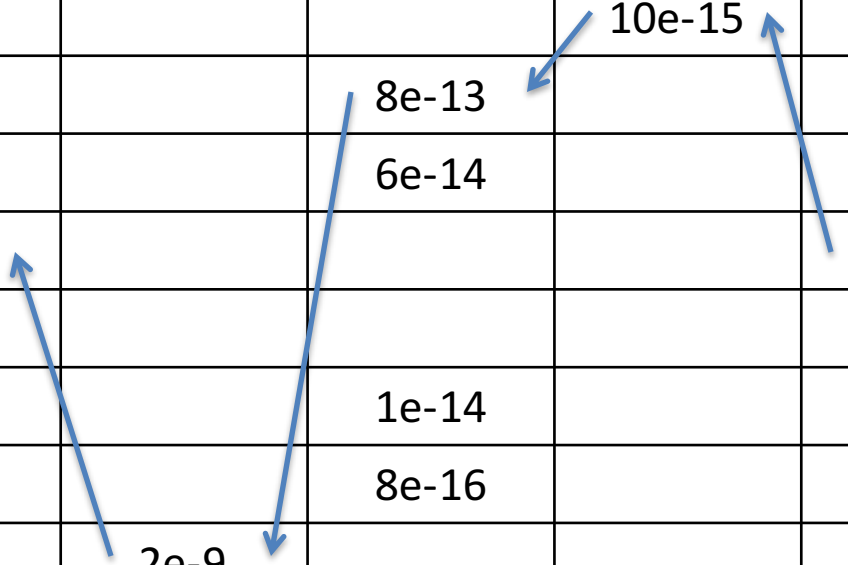
- In general, $s_{i-1} = bp_i(s_i)$.

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

	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	

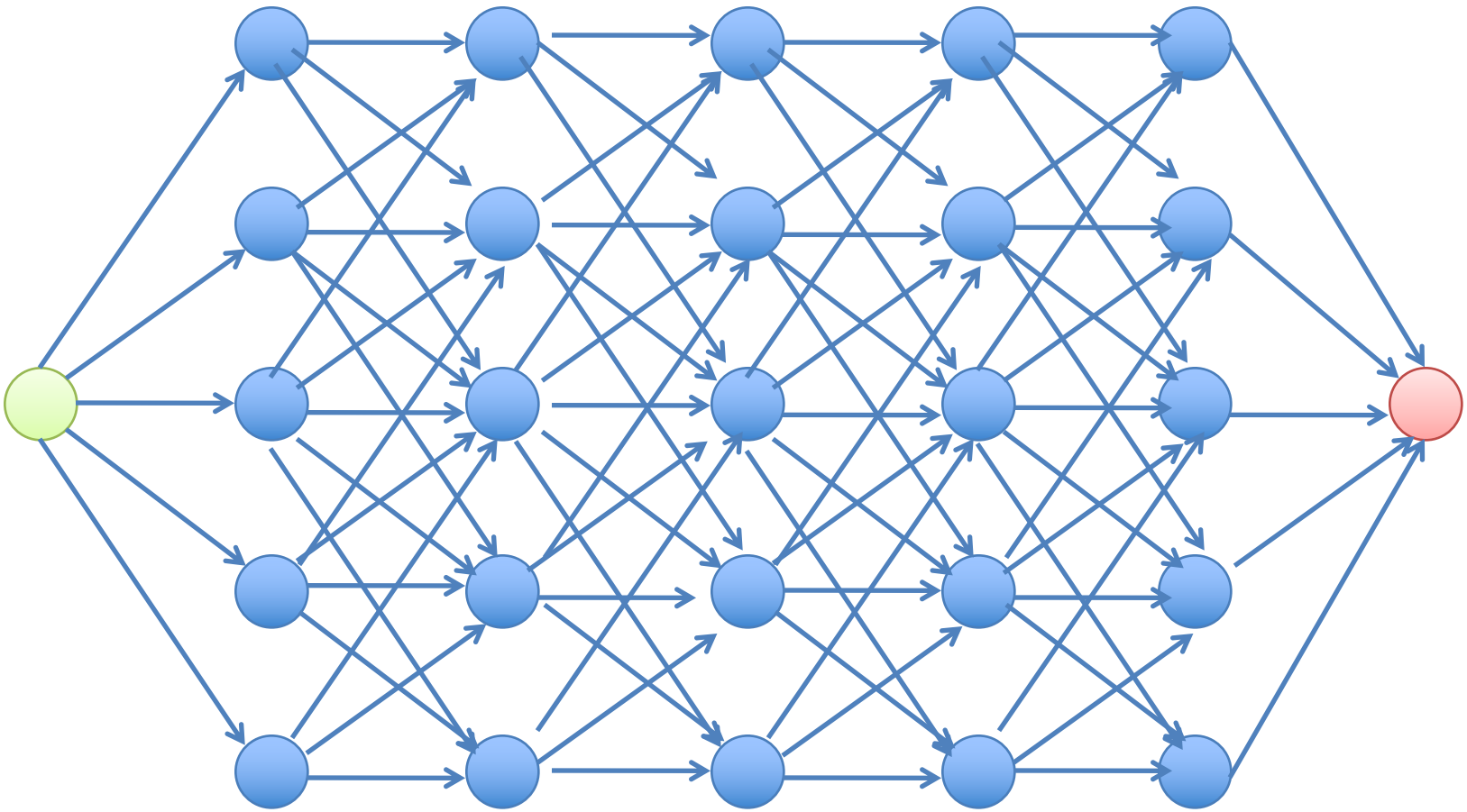


General Idea: Dynamic Programming

动态规划

- Use a table data structure to store partial quantities that will be reused many times.
  - Optimal substructure: best solution to a problem relies on best solutions to its (similar-looking) subproblems.
  - Overlapping subproblems: reuse a small number of quantities many times
- Examples: Viterbi, minimum Levenshtein distance, Dijkstra's shortest path algorithm, …

A Different View: Best Path



Asymptotic Analysis

Memory:

- The table is $n \times |\Lambda|$.

Runtime:

- Each cell in the table requires $O(|\Lambda|)$ operations.
- Total runtime is $O(n|\Lambda|^2)$.

Lecture Outline

- ✓ Markov models
- ✓ Hidden Markov models
- ✓ Viterbi algorithm
4. Other inference algorithms for HMMs
5. Learning algorithms for HMMs

COFFEE BREAK

Viterbi Algorithm (Recap)

- 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:

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

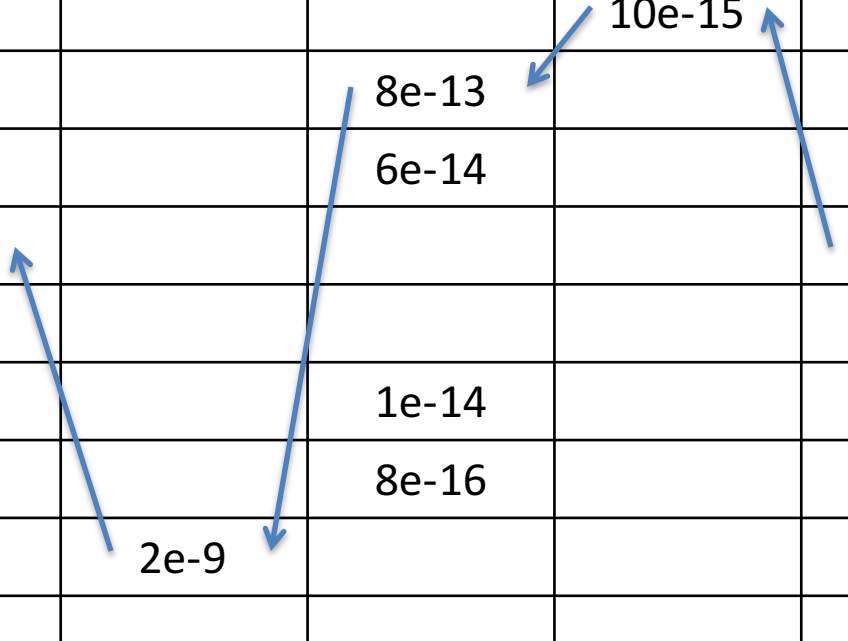
$$\text{bp}_n(\sigma) = \arg\max_{\sigma' \in \Lambda} \gamma(\sigma \mid \sigma') \times \text{score}_{n-1}(\sigma')$$

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	

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	



Lecture Outline

- ✓ Markov models
- ✓ Hidden Markov models
- ✓ Viterbi algorithm
4. Other inference algorithms for HMMs
5. Learning algorithms for HMMs

How To Calculate …

Given the HMM and a sequence:

✓ 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 (**γ** and **η**)?

Problem 2: $p(\mathbf{w} \mid \boldsymbol{\gamma}, \boldsymbol{\eta})$

- Why might we be interested in this quantity?
  - Using an HMM as a language model, we might want to compare two or more sequences.
  - Later, we will want to *maximize* this quantity with respect to the parameters $\boldsymbol{\gamma}$ and $\boldsymbol{\eta}$ (learning).

Maximizing and Summing

Most probable state sequence given words:

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

Combinatorial optimization problem, solvable in polynomial time.

Total probability of all state sequences, together with words:

$$p(\boldsymbol{w} \mid \boldsymbol{\gamma}, \boldsymbol{\eta}) = \sum_{\boldsymbol{s}} p(\boldsymbol{s}, \boldsymbol{w} \mid \boldsymbol{\gamma}, \boldsymbol{\eta})$$

产生w序列
n个位置的所有状态组合
的score和

A Very Similar Trick

- The sum of all label-sequence probabilities breaks down into the sum over scores for different final symbols.

$$\sum_{\sigma \in \Lambda} \underbrace{\sum_{s_1 \ldots s_{n-1}} p(s_1 \ldots s_{n-1}\sigma, \boldsymbol{w} \mid \boldsymbol{\gamma}, \boldsymbol{\eta})}_{f_n(\sigma)}$$

产生w序列
位置n取状态delta
前n-1个位置取各种状态组合
的score和

A Very Similar Trick

- As before, there is a recurrence.
- Here, we exploit the fact that multiplication distributes over addition.

产生w序列的前n-1个w
位置n-1取状态delta'
前n-2个位置取各种状态组合
的score和

$$\sum_{\sigma \in \Lambda} \underbrace{\sum_{s_1 \ldots s_{n-1}} p(s_1 \ldots s_{n-1} \sigma, \boldsymbol{w} \mid \boldsymbol{\gamma}, \boldsymbol{\eta})}_{f_n(\sigma)}$$

$$= \sum_{\sigma \in \Lambda} \gamma(\text{stop} \mid \sigma) \times \eta(w_n \mid \sigma) \sum_{\sigma' \in \Lambda} \underbrace{\sum_{s_1 \ldots s_{n-2}} p(s_1 \ldots s_{n-2} \sigma', w_1 \ldots w_{n-1} \mid \boldsymbol{\gamma}, \boldsymbol{\eta})}_{f_{n-1}(\sigma')}$$

Forward Algorithm

产生w序列的前n-1个w
位置n-1取状态delta'
前n-2个位置取各种状态组合
的score和

- For every σ in Λ, let:

$$f_1(\sigma) = \eta(w_1 \mid \sigma) \times \gamma(\sigma \mid \text{start})$$

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

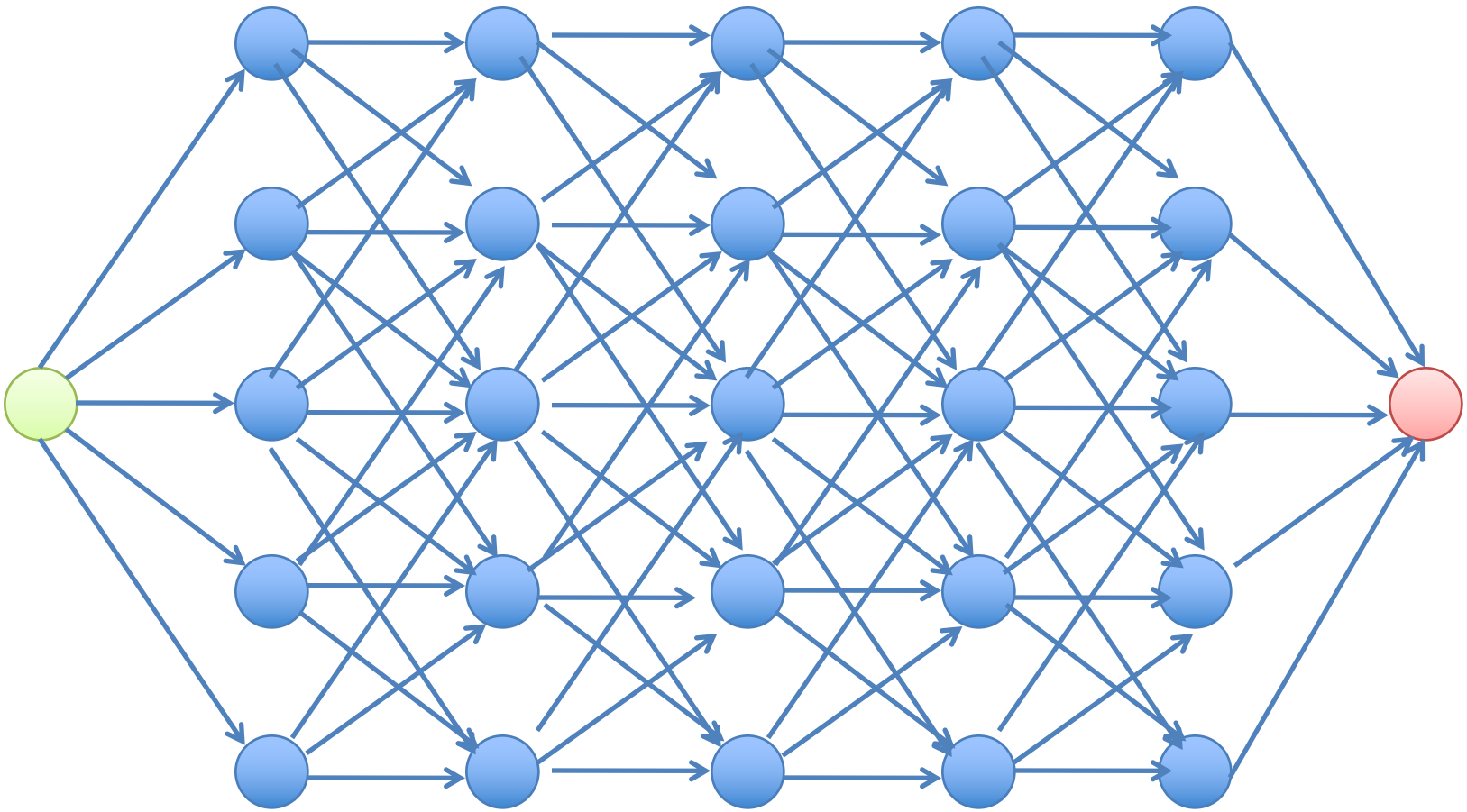
$$f_i(\sigma) = \eta(w_i \mid \sigma) \times \sum_{\sigma' \in \Lambda} \gamma(\sigma \mid \sigma') \times f_{i-1}(\sigma')$$

- For every σ in Λ:

$$f_n(\sigma) = \gamma(\text{stop} \mid \sigma) \times \eta(w_n \mid \sigma) \times \sum_{\sigma' \in \Lambda} \gamma(\sigma \mid \sigma') \times f_{n-1}(\sigma')$$

$$p(\boldsymbol{w} \mid \boldsymbol{\gamma}, \boldsymbol{\eta}) = \sum_{\sigma \in \Lambda} f_n(\sigma)$$

A Different View: Path Sum



A Different View: Linear System

- |Λ| times n free variables, same number of equations.
- Can rewrite as a matrix inversion problem!

$$
\begin{aligned}
f_1(\sigma) &= \eta(w_1 \mid \sigma) \times \gamma(\sigma \mid \text{start}) \\
f_i(\sigma) &= \eta(w_i \mid \sigma) \times \sum_{\sigma' \in \Lambda} \gamma(\sigma \mid \sigma') \times f_{i-1}(\sigma') \\
f_n(\sigma) &= \gamma(\text{stop} \mid \sigma) \times \eta(w_n \mid \sigma) \times \sum_{\sigma' \in \Lambda} \gamma(\sigma \mid \sigma') \times f_{n-1}(\sigma')
\end{aligned}
$$

From Forward to Backward

- Forward algorithm: precomputation of partial sums, from i = 1 to n, each involving $|\Lambda|$ quantities, each a sum over $|\Lambda|$ combinations.
  - Asymptotic analysis is the same as Viterbi.
- No need to start at the left and move right!
- **Backward algorithm** calculates partial sums from the right to the left.

Backward Algorithm

从i开始，出wi到wn
所有状态组合的概率的和

- For every σ in Λ, let:

$$b_n(\sigma) \quad = \quad \gamma(\text{stop} \mid \sigma) \times \eta(w_n \mid \sigma)$$

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

$$b_i(\sigma) \quad = \quad \eta(w_i \mid \sigma) \times \sum_{\sigma' \in \Lambda} \gamma(\sigma' \mid \sigma) \times b_{i+1}(\sigma')$$

- For every σ in Λ:

$$b_1(\sigma) = \gamma(\sigma \mid \text{start}) \times \eta(w_1 \mid \sigma) \sum_{\sigma' \in \Lambda} \gamma(\sigma' \mid \sigma) \times b_2(\sigma')$$

$$p(\boldsymbol{w} \mid \boldsymbol{\gamma}, \boldsymbol{\eta}) \quad = \quad \sum_{\sigma \in \Lambda} b_1(\sigma)$$

Forward and Backward

- Two different ways to rearrange the sums of products of sums of products of sums of products.
- Different intermediate quantities.

Pop Quiz

- Might we have done the same thing with the Viterbi algorithm?

	maximize and multiply operations	add and multiply operations
works left to right	Viterbi	Forward
works right to left	?	Backward

Generalization: Semirings

- Viterbi and Forward algorithms correspond to exactly the same calculations, except one maximizes and the other sums.
- One view: they are the *same* abstract algorithm, instantiated in two different **semirings**.
- Informally, a semiring is a set of values and some operations that obey certain properties.

半环：抽象代数中，类似于环但没有加法逆元的代数结构

Semirings, More Formally

- A set of values, including a “zero” (additive identity and multiplicative annihilator) and a “one” (multiplicative identity).
- Two operations: “plus” and “times.”
  - “Plus” is associative and commutative.
  - “Times” is associative.
- “Times” distributes over “plus.”
  - This is what we have exploited to get efficient algorithms for maximizing and summing!

Semirings

	Real	Viterbi
set of values	nonnegative reals	nonnegative reals
“zero”	0	0
“one”	1	1
“plus”	+	max
“times”	×	×

Some Other Semirings

- **Boolean**: use to determine whether the HMM can produce the string at all.
- **Counting**: use to determine how many valid labelings there are.
  - Could be less than $|\Lambda|^n$, if some transition and/or emission probabilities are zero.
- **Log-real**: use with log-probabilities to avoid underflow.
- **K-best**: use to find the K best label sequences.
- **Min-cost**: used with Levenshtein edit distance and Dijkstra's algorithms.

How To Calculate …

Given the HMM and a sequence:

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

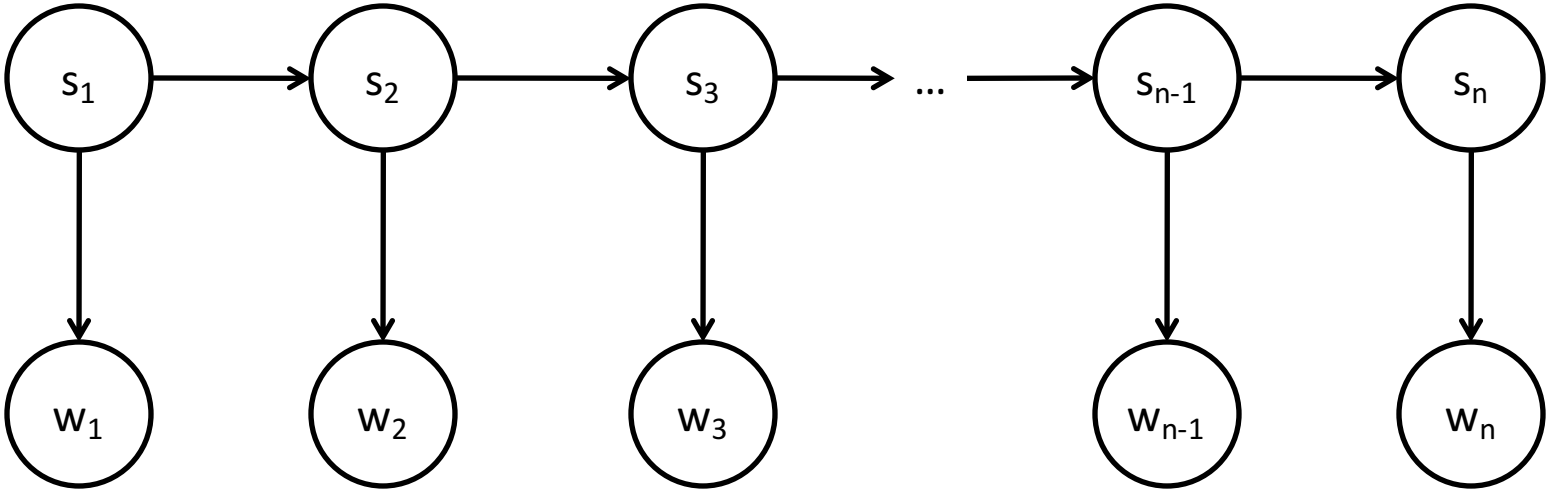
Given states and sequences, or just states:

5. The parameters of the HMM (**γ** and **η**)?

Random Variables

- So far we've focused on reasoning about the *whole sequence* of states.
- Local reasoning was in service of:
  - Finding the best **s** = s_1 s_2 … s_n
  - Finding the total probability of **w**, averaging over all possible values of **s**.
- It is helpful to use a graphical representation of all our random variables.

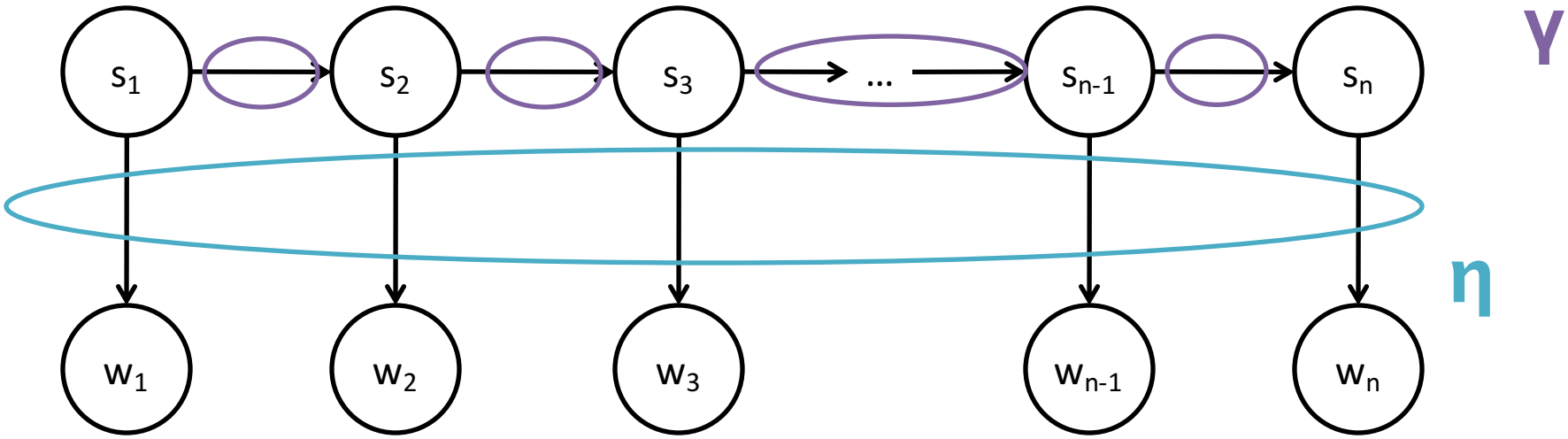
Graphical Model Representation



- Each node is a random variable taking some value.
- Incoming edges to a r.v. tell what other r.v.s it conditions on directly.

$$p(\boldsymbol{x}) \quad = \quad \prod_i p(x_i \mid \text{parents}(x_i))$$

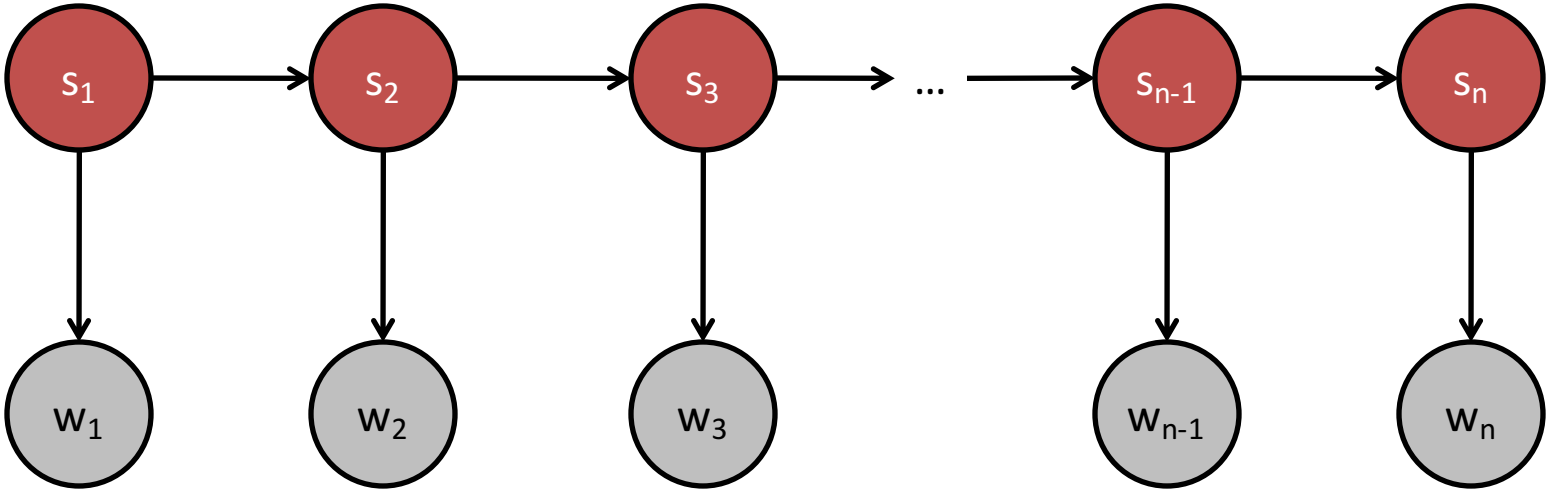
Graphical Model Representation



- Each node is a random variable taking some value.
- Incoming edges to a r.v. tell what other r.v.s it conditions on directly.

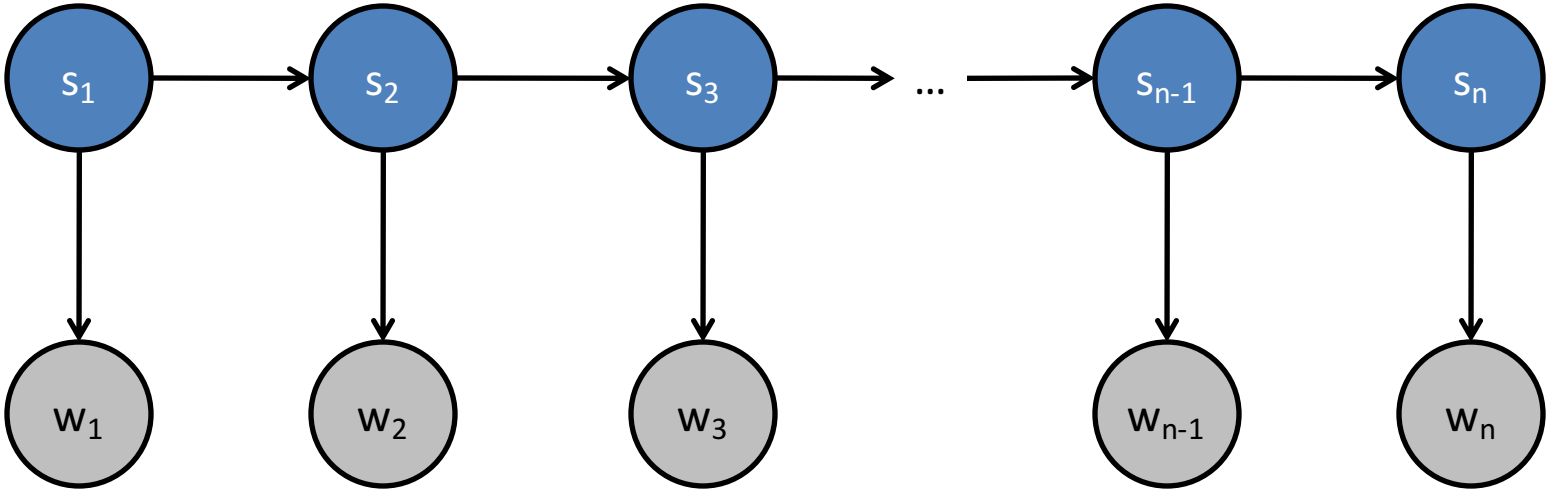
$$p(\boldsymbol{x}) = \prod_i p(x_i \mid \text{parents}(x_i))$$

Problem 1: Most Likely **s**



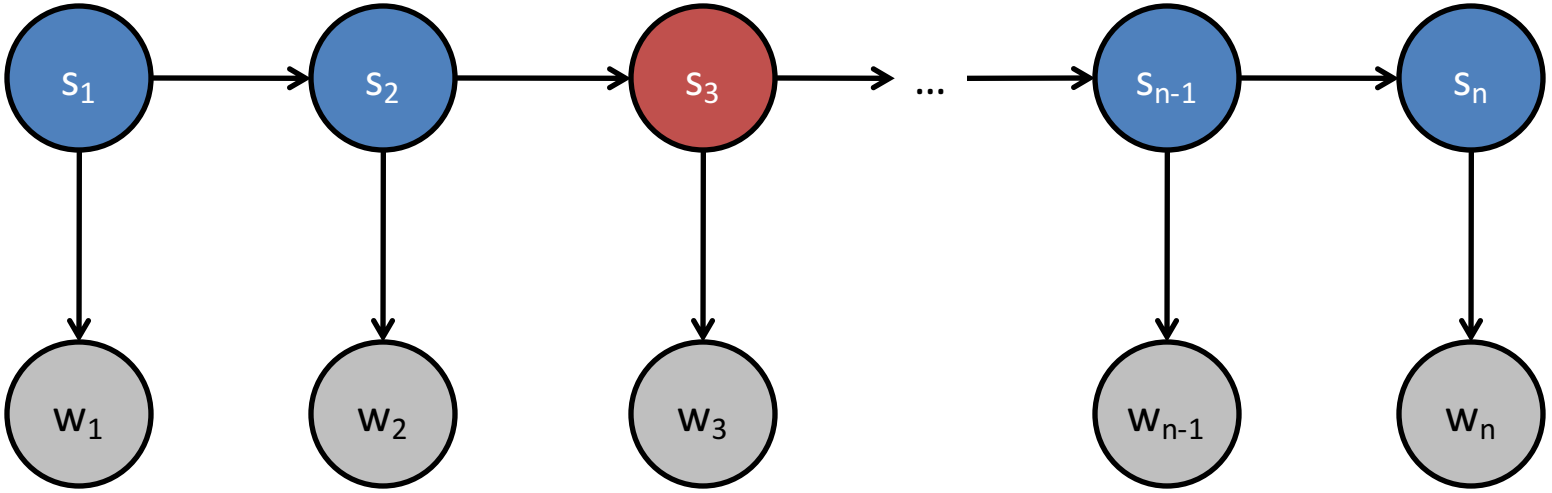
- Sequence of words is *observed*, so we color those r.v.s gray.
- We want to assign values, collectively, to the states, so we color those red.
- Goal: calculate the best value of p for any assignment to "red" r.v.s., respecting "gray" evidence r.v.s.

Problem 2: Probability of **w**



- Sequence of words is *observed*, so we color those r.v.s gray.
- We want to sum over all settings of the state r.v.s, so we color them blue.
- Goal: calculate the best value of p for any assignment to "red" r.v.s, respecting "gray" evidence r.v.s and summing over all possible assignments to "blue" r.v.s.
  - There are no red r.v.s in this problem!

Problem 3: A Single s_i

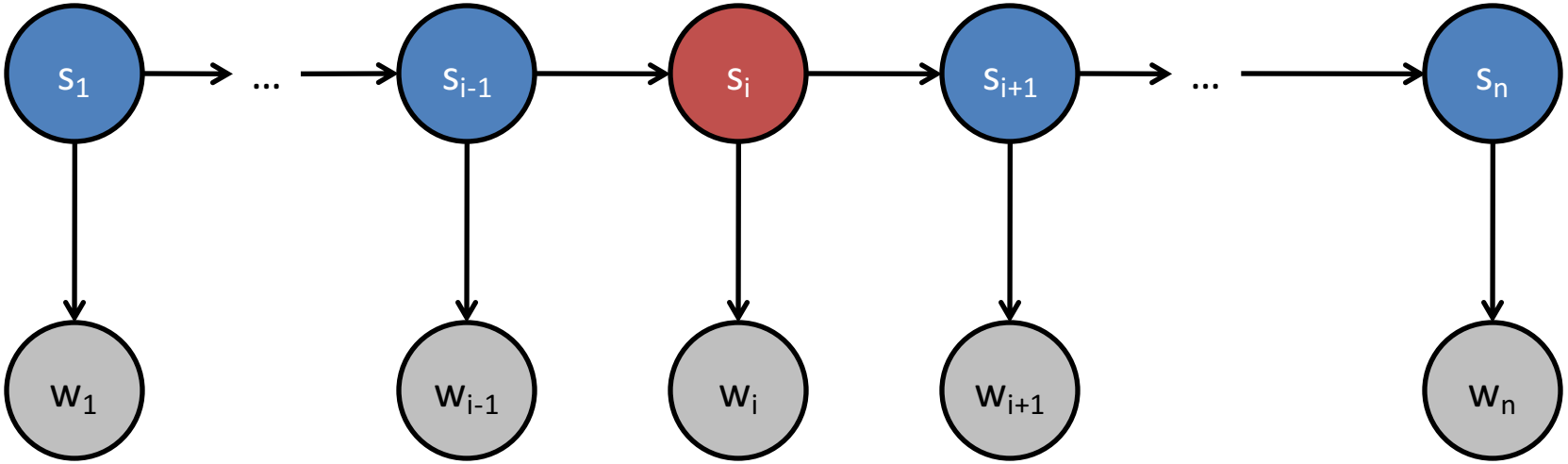


- Goal: calculate the best value of p for any assignment to "red" r.v.s., respecting "gray" evidence r.v.s. and summing over all possible assignments to "blue" r.v.s.

Aside

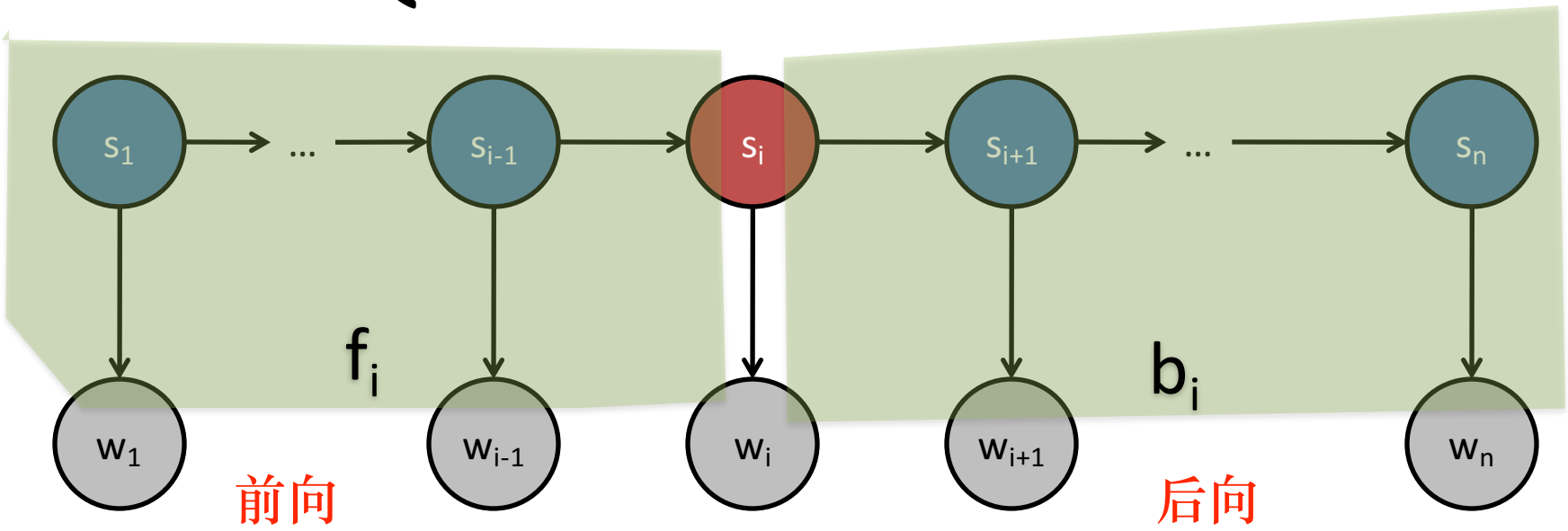
- Probabilistic graphical models are an extremely useful framework for machine learning in NLP and other complex problems.
- HMMs are one kind of graphical model; there are many others you may have heard of.
  - Bayesian networks
  - Markov networks
  - Factor graphs

Quantities We Need



$$
\begin{aligned}
p(\boldsymbol{w}, s_i \mid \boldsymbol{\gamma}, \boldsymbol{\eta}) &= \sum_{s_1 \ldots s_{i-1}} \sum_{s_{i+1} \ldots s_n} p(s_1 \ldots s_i \ldots s_n, \boldsymbol{w} \mid \boldsymbol{\gamma}, \boldsymbol{\eta}) \\
&= \sum_{s_1 \ldots s_{i-1}} \sum_{s_{i+1} \ldots s_n} p(s_1 \ldots s_i, w_1 \ldots w_i \mid \boldsymbol{\gamma}, \boldsymbol{\eta}) \times p(s_{i+1} \ldots s_n, w_{i+1} \ldots w_n \mid s_i, \boldsymbol{\gamma}, \boldsymbol{\eta}) \\
&= \sum_{s_1 \ldots s_{i-1}} p(s_1 \ldots s_i, w_1 \ldots w_i \mid \boldsymbol{\gamma}, \boldsymbol{\eta}) \sum_{s_{i+1} \ldots s_n} p(s_{i+1} \ldots s_n, w_{i+1} \ldots w_n \mid s_i, \boldsymbol{\gamma}, \boldsymbol{\eta}) \\
&= f_i(s_i) \times b_i(s_i)
\end{aligned}
$$

Quantities We Need



$$
\begin{aligned}
p(\boldsymbol{w}, s_i \mid \boldsymbol{\gamma}, \boldsymbol{\eta}) &= \sum_{s_1 \ldots s_{i-1}} \sum_{s_{i+1} \ldots s_n} p(s_1 \ldots s_i \ldots s_n, \boldsymbol{w} \mid \boldsymbol{\gamma}, \boldsymbol{\eta}) \\
&= \sum_{s_1 \ldots s_{i-1}} \sum_{s_{i+1} \ldots s_n} p(s_1 \ldots s_i, w_1 \ldots w_i \mid \boldsymbol{\gamma}, \boldsymbol{\eta}) \times p(s_{i+1} \ldots s_n, w_{i+1} \ldots w_n \mid s_i, \boldsymbol{\gamma}, \boldsymbol{\eta}) \\
&= \sum_{s_1 \ldots s_{i-1}} p(s_1 \ldots s_i, w_1 \ldots w_i \mid \boldsymbol{\gamma}, \boldsymbol{\eta}) \sum_{s_{i+1} \ldots s_n} p(s_{i+1} \ldots s_n, w_{i+1} \ldots w_n \mid s_i, \boldsymbol{\gamma}, \boldsymbol{\eta}) \\
&= f_i(s_i) \times b_i(s_i)
\end{aligned}
$$

Distribution over States for w_i

- Run forward and backward algorithms to produce f_i and b_i.
  - If we only care about one state, we can stop at i.
- For each σ in Λ:

$$p(S_i = \sigma, \boldsymbol{w} \mid \boldsymbol{\gamma}, \boldsymbol{\eta}) = f_i(\sigma) \times b_i(\sigma)$$

$$p(S_i = \sigma \mid \boldsymbol{w}, \boldsymbol{\gamma}, \boldsymbol{\eta}) = \frac{p(S_i = \sigma, \boldsymbol{w} \mid \boldsymbol{\gamma}, \boldsymbol{\eta})}{\sum_{\sigma' \in \Lambda} p(S_i = \sigma', \boldsymbol{w} \mid \boldsymbol{\gamma}, \boldsymbol{\eta})}$$

"posterior"

- Note that the denominator is $p(\mathbf{w} \mid \boldsymbol{\gamma}, \boldsymbol{\eta})$.

How To Calculate …

Given the HMM and a sequence:

- ✓ The most probable state sequence?
- ✓ The probability of the word sequence?
- ✓ 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 (**γ** and **η**)?

Building on Per-Word Posteriors

- Total runtime for forward and backward algorithms is $O(n|\Lambda|^2)$.
- Once you have all f and b quantities, you can calculate the posteriors for every word's label.

$$\hat{s}_i \quad \leftarrow \quad \arg\max_{\sigma \in \Lambda} \underbrace{p(S_i = \sigma \mid \boldsymbol{w}, \boldsymbol{\gamma}, \boldsymbol{\eta})}_{f_i(\sigma) \times b_i(\sigma)}$$

- This is sometimes called **posterior decoding**.

Posterior Decoding

- This approach to decoding exploits the full distribution over sequences to choose each word's label. For each i:

$$\hat{s}_i \quad \leftarrow \quad \arg\max_{\sigma \in \Lambda} \underbrace{p(S_i = \sigma \mid \boldsymbol{w}, \boldsymbol{\gamma}, \boldsymbol{\eta})}_{f_i(\sigma) \times b_i(\sigma)}$$

- Compare with MAP decoding (sometimes called "Viterbi" decoding after the algorithm that accomplishes it:

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

Which One to Use?

- They will not, in general, give the same label sequence.
- Sometimes one works better, sometimes the other.
- Posterior decoding can give a label sequence that itself gets *zero* probability!
- There is a way to unify both.

Cost

- Imagine that, once we construct our HMM, we are going to play a game.
- The HMM will be given a new sequence **w**.
- We must label **w** using the HMM.
- Our label sequence **s** will be compared to the true one, **s***.
- Depending on how badly we do, we will pay a fine.
- We want to minimize the cost.
- Without seeing **w**, our strategy will depend on how the cost is defined!

All-or-Nothing Cost

- Suppose we will pay 1€ if we get the sequence wrong, i.e., if $\mathbf{s} \neq \mathbf{s}^*$.
- Otherwise we pay nothing.

- What should we do?
- If we trust our distribution $p(\mathbf{w}, \mathbf{s} \mid \boldsymbol{\gamma}, \boldsymbol{\eta})$, then we should use the most probable whole-sequence $\mathbf{s}$.
  - Viterbi

Hamming Cost

- Alternately, suppose we pay 0.10 € for every *word* that we label incorrectly.
- This is more forgiving, and suggests that we focus on reasoning about each word without worrying about the coherence of the whole sequence.

- What should we do?
- If we trust our distribution $p(\mathbf{w}, \mathbf{s} \mid \boldsymbol{\gamma}, \boldsymbol{\eta})$, then we should use the most label for each word.
  - Posterior decoding

Minimum Bayes Risk

- The assumption that we have a good estimated distribution p(**w**, **s** |**γ**, **η**) leads naturally to the following decoding rule:

$$\hat{\boldsymbol{s}} \quad \leftarrow \quad \arg\min_{\boldsymbol{s}'} \mathbb{E}_{p(\boldsymbol{s}|\boldsymbol{w},\boldsymbol{\gamma},\boldsymbol{\eta})}[\mathrm{cost}(\boldsymbol{s}', \boldsymbol{s})]$$

- Pick the **s** that is least offensive, in expectation.
  - With all-or-nothing cost, we get MAP/Viterbi decoding.
  - With Hamming cost, we get posterior decoding.

Word-Wise Costs

- If the cost function is a sum of local costs, we can exploit linearity of expectation:

$$\begin{aligned}\mathbb{E}_{p(\boldsymbol{s}|\boldsymbol{w},\boldsymbol{\gamma},\boldsymbol{\eta})}[\mathrm{cost}(\boldsymbol{s}',\boldsymbol{s})] &= \mathbb{E}_{p(\boldsymbol{s}|\boldsymbol{w},\boldsymbol{\gamma},\boldsymbol{\eta})}\left[\sum_i \mathrm{cost}_i(s'_i, s_i)\right] \\ &= \sum_i \mathbb{E}_{p(\boldsymbol{s}|\boldsymbol{w},\boldsymbol{\gamma},\boldsymbol{\eta})}[\mathrm{cost}_i(s'_i, s_i)]\end{aligned}$$

- For the Hamming cost, we can make independent decisions once we know the expected local costs.

Manipulating the Cost Function

- Suppose we have an HMM for named entity recognition.
  - Tags are B, I, and O.
- If we really care about precision (finding only correct named entities, at risk of missing some), what cost function makes sense?
- What if we really care about *recall*?

One More Cost Function

- BIO tagging again.
- Suppose we assign different costs to recall, precision, and *boundary* errors.

	correct:	B-B	B-I	B-O	I-B	I-I	I-O	O-B	O-O
hyp.:	B-B		split	prec.		split	prec.		prec.
	B-I	merge		bound.	merge		bound.	bound.	bound.
	B-O	recall	recall		recall	bound.		recall	
	I-B		split	prec.		split	prec.		prec.
	I-I	merge		bound.	merge		bound.	bound.	bound.
	I-O	recall	recall		recall	bound.		recall	
	O-B		prec.	prec.		bound.	prec.		prec.
	O-O	recall			recall	recall		recall	

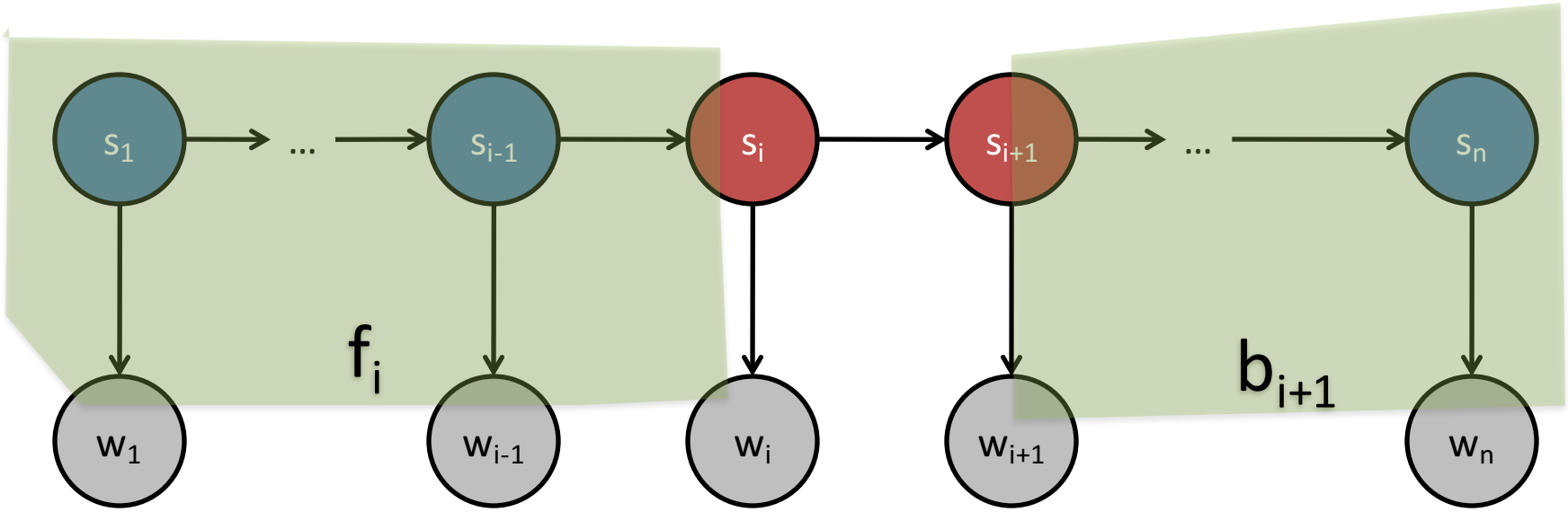
A More Complex Posterior

$$p(s_i s_{i+1} \mid \boldsymbol{w}, \boldsymbol{\gamma}, \boldsymbol{\eta})$$

- Use the same trick as before, but pair up slightly different forward and backward probabilities:

$$\begin{aligned}
p(s_i s_{i+1}, \boldsymbol{w} \mid \boldsymbol{\gamma}, \boldsymbol{\eta}) &= \sum_{s_1 \ldots s_{i-1}} \sum_{s_{i+2} \ldots s_n} p(s_1 \ldots s_i s_{i+1} \ldots s_n, \boldsymbol{w} \mid \boldsymbol{\gamma}, \boldsymbol{\eta}) \\
&= \sum_{s_1 \ldots s_{i-1}} p(s_1 \ldots s_i, w_1 \ldots w_i \mid \boldsymbol{\gamma}, \boldsymbol{\eta}) \times \gamma(s_{i+1} \mid s_i) \times \eta(w_{i+1} \mid s_{i+1}) \\
&\quad \times \sum_{s_{i+2} \ldots s_n} p(s_{i+2} \ldots s_n, w_{i+1} \ldots w_n \mid s_{i+1}, \boldsymbol{\gamma}, \boldsymbol{\eta}) \\
&= f_i(s_i) \times \gamma(s_{i+1} \mid s_i) \times \eta(w_{i+1} \mid s_{i+1}) \times b_{i+1}(s_{i+1})
\end{aligned}$$

Pairwise Posterior



$$
\begin{aligned}
p(s_i s_{i+1}, \boldsymbol{w} \mid \boldsymbol{\gamma}, \boldsymbol{\eta}) &= \sum_{s_1 \ldots s_{i-1}} \sum_{s_{i+2} \ldots s_n} p(s_1 \ldots s_i s_{i+1} \ldots s_n, \boldsymbol{w} \mid \boldsymbol{\gamma}, \boldsymbol{\eta}) \\
&= \sum_{s_1 \ldots s_{i-1}} p(s_1 \ldots s_i, w_1 \ldots w_i \mid \boldsymbol{\gamma}, \boldsymbol{\eta}) \times \gamma(s_{i+1} \mid s_i) \times \eta(w_{i+1} \mid s_{i+1}) \\
&\quad \times \sum_{s_{i+2} \ldots s_n} p(s_{i+2} \ldots s_n, w_{i+1} \ldots w_n \mid s_{i+1}, \boldsymbol{\gamma}, \boldsymbol{\eta}) \\
&= f_i(s_i) \times \gamma(s_{i+1} \mid s_i) \times \eta(w_{i+1} \mid s_{i+1}) \times b_{i+1}(s_{i+1})
\end{aligned}
$$

A Problem

- We can label each adjacent pair of words, but nothing guarantees that our labels will be consistent with each other.

Pop Quiz

1. Can you think of a cost function such that minimum Bayes risk decoding *can't* be done in polynomial time?

Quiz Answer (1)

- Every word must get a different label.
  - Equivalently, we can use each label at most once.
- Each label must be used exactly once.
  - Hamiltonian path.

Pop Quiz

Part 2:

- We want to choose a *single* sequence of labels to minimize risk.
- The scoring function factors into local scores (sum of pairwise posteriors).
- How do we do it?

最短路径

Quiz Answer (2)

- Pairwise local costs are kind of like transition probabilities.
  - Except we want to *minimize* their *sum* rather than maximize their product.
- Viterbi algorithm, but with min-cost semiring, a different "transition" score, and no "emission" score.

Pairwise Minimum Bayes Risk

- First, run forward and backward.
- Use posteriors to score all possible label-pairs for all adjacent words.
- Use a Viterbi-like algorithm to find the best-scoring, consistent labeling.
  - Use min-cost semiring.
  - Algorithm scores label pairs by local cost times posterior probability (local risk).

How To Calculate …

Given the HMM and a sequence:

- ✓ The most probable state sequence?
- ✓ The probability of the word sequence?
- ✓ The probability distribution over states, for each word?
- ✓ Minimum risk sequence

Given states and sequences, or just states:

5. The parameters of the HMM (**γ** and **η**)?

Lecture Outline

- ✓ Markov models
- ✓ Hidden Markov models
- ✓ Viterbi algorithm
- ✓ Other inference algorithms for HMMs
5. Learning algorithms for HMMs

Learning HMMs

- Typical starting point: we have some data to learn from, and we know how many states the HMM has.
  - We may also have constraints on the states, but assume for now that we do not.
- Two main possibilities:
  - Supervised: we have **complete** data: example pairs (**w**, **s**).
  - Unsupervised: we only have examples of **w**.

Supervised Learning of HMMs

- The building blocks of HMMs are multinomial distributions
  - **γ**: distribution over next state given current state
  - **η**: distribution over word given current state
- Statistics offers us the **maximum likelihood** principle:

$$\langle \hat{\gamma}, \hat{\eta} \rangle \quad \leftarrow \quad \arg\max_{\langle \gamma, \eta \rangle} p(\boldsymbol{s}, \boldsymbol{w} \mid \gamma, \eta)$$

Separability of Learning

- With observed data, each state's transition and emission distributions can be estimated *separately*
  - from each other
  - from those of all other states
- The result is that learning is very simple and fast.

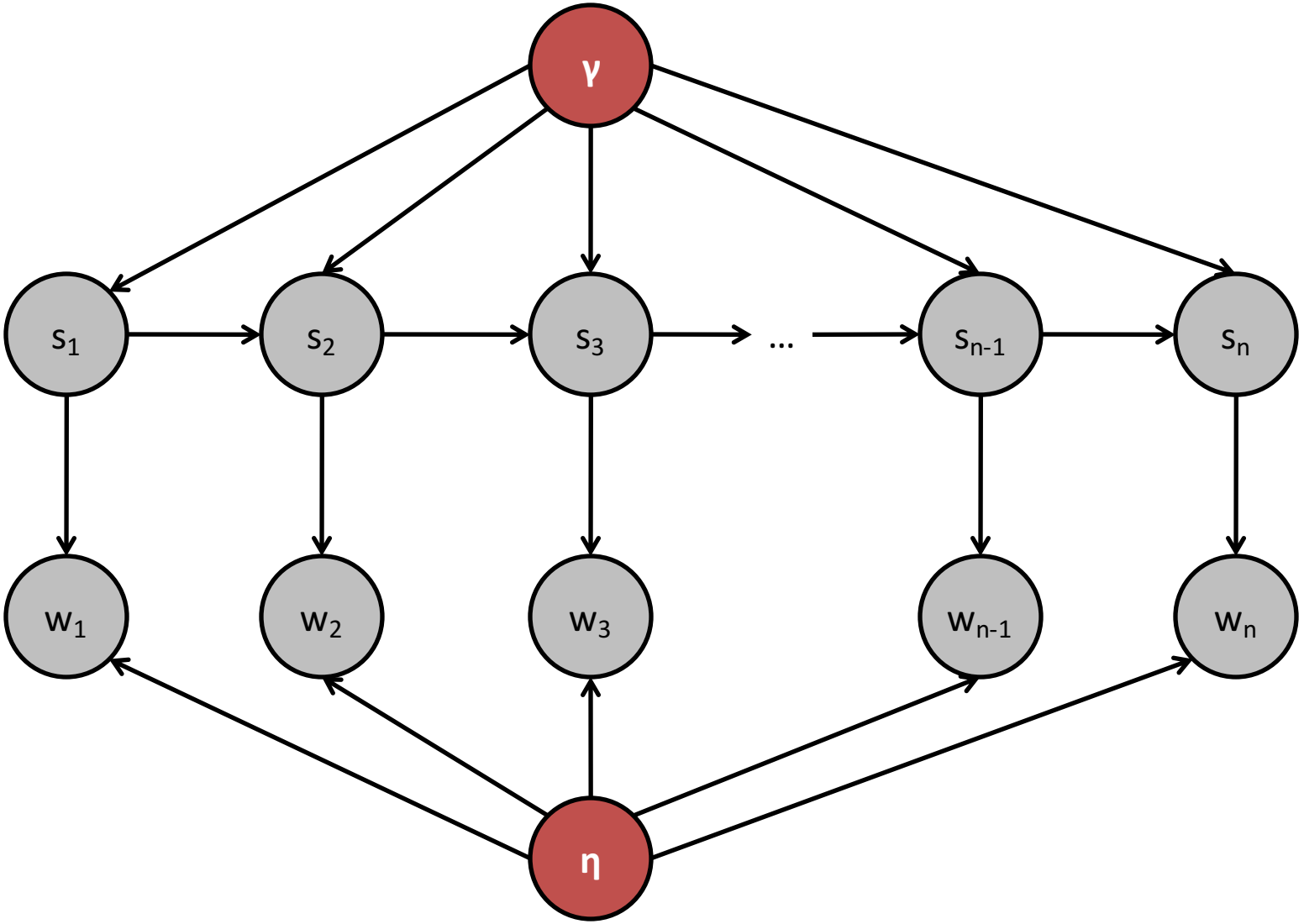
MLE by Relative Frequencies

- (I'm skipping the derivation; it involves log likelihood, a Lagrangian multiplier, and some differential calculus.)

$$\hat{\gamma}(\sigma \mid \sigma') = \frac{\text{freq}(\sigma'\sigma)}{\text{freq}(\sigma)}$$

$$\hat{\eta}(w \mid \sigma) = \frac{\text{freq}\begin{pmatrix} \sigma \\ w \end{pmatrix}}{\text{freq}(\sigma)}$$

Graphical Models View



Learning with a Prior

- The techniques described so far are examples of **maximum likelihood estimation** (MLE).
- MLE works well when there is a lot of data.
  - We never have as much as we'd like.
- Learning with a prior is a way to inject some background knowledge and avoid overfitting to the training data.
- MAP learning:

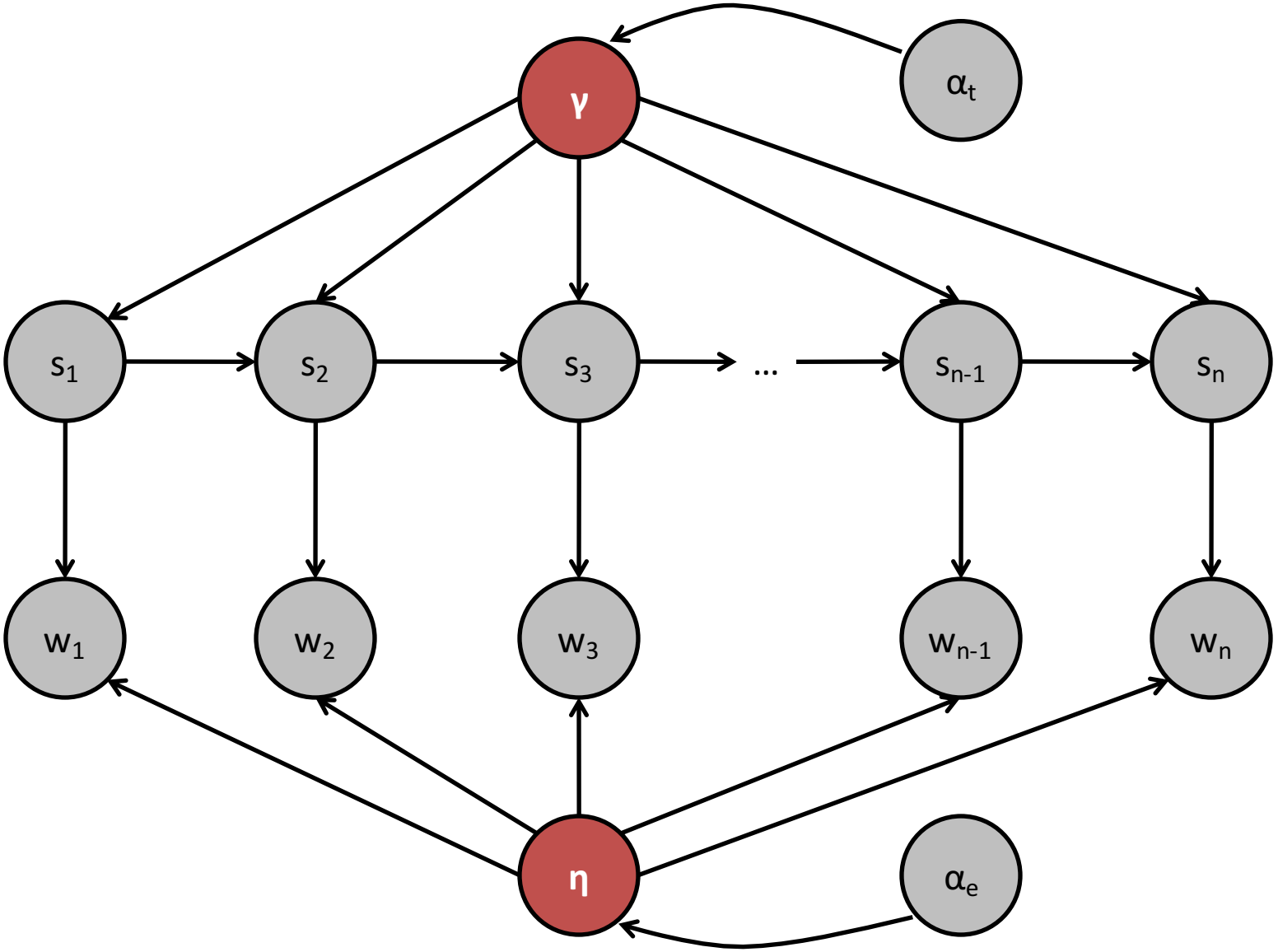
$$\langle \hat{\boldsymbol{\gamma}}, \hat{\boldsymbol{\eta}} \rangle \quad \leftarrow \quad \arg\max_{\langle \boldsymbol{\gamma}, \boldsymbol{\eta} \rangle} p(\boldsymbol{s}, \boldsymbol{w} \mid \boldsymbol{\gamma}, \boldsymbol{\eta}) \times p(\boldsymbol{\gamma}, \boldsymbol{\eta})$$

this part is new

Priors for HMMs

- HMMs are built out of multinomial distributions.
- An easy prior for the multinomial distribution is the Dirichlet distribution.
  - Simplest version: *symmetric,* non-sparse Dirichlet.
- In practice:
  - Choose $\alpha_t > 1$ for transitions and $\alpha_e > 1$ for emissions.
  - Before normalizing frequencies, add $\alpha_t - 1$ to transition counts and $\alpha_e - 1$ to emission counts.
  - Exactly the same as additive smoothing in language models.

Graphical Models View



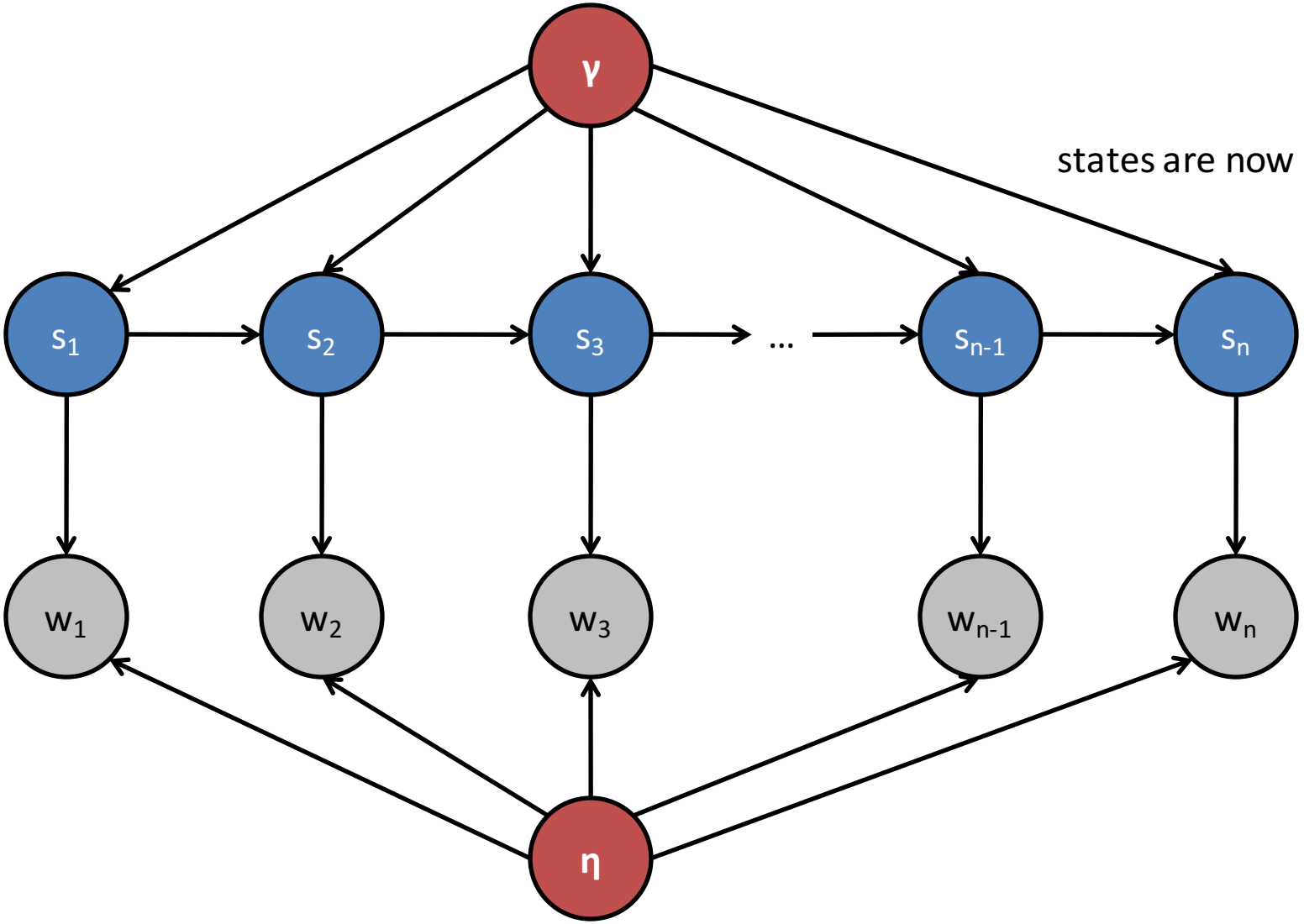
Unsupervised Learning of HMMs

- Historically (going back to the 1960s), we do not observe the states, even during training time.
  - Hence the name: *hidden* Markov models.
- The earliest instance of a parameter estimation problem where the data are incomplete.
- How do we learn only from **w**?

We Already Have All The Tools!

- Statistics: maximum likelihood estimation.
  - Counting events and normalizing the counts.
- Computation: inference about posteriors.
  - Forward and backward algorithms.

Graphical Models View

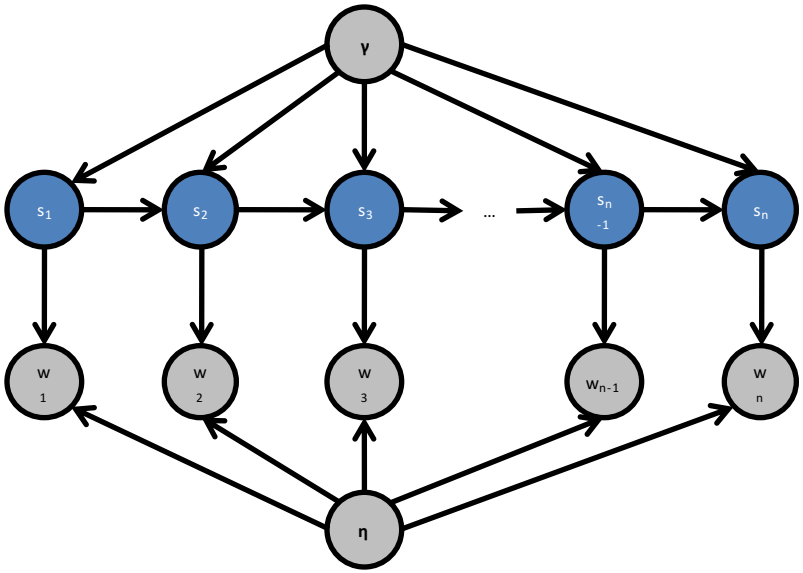


Mixed Inference

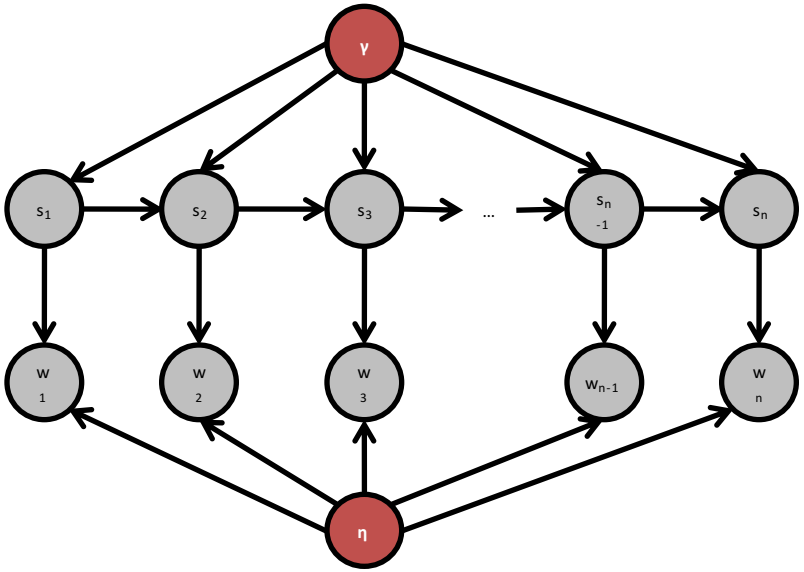
- Mixed inference problems – where we want to sum out some random variables while getting values for others – are hard in general.
- Indeed, finding the MLE is an NP-hard problem!
- Optimization view: we are optimizing a non-convex function (of the parameters).

High-Level View of EM

- EM stands for “expectation maximization.”



E step: infer posterior distribution over missing data.



M step: maximum likelihood estimation with soft values for missing data.

Procedural View of EM

- Begin with an initial estimate of the parameters of the HMM ($\boldsymbol{\gamma}$ and $\boldsymbol{\eta}$).
- Iterate:
  - E step: calculate probability of each possible transition and each possible emission at each position.
  - M step: re-estimate parameters to maximize likelihood of "complete" data.

E Step

- Calculate $p(\mathbf{s} \mid \mathbf{w}, \boldsymbol{\gamma}, \boldsymbol{\eta})$ for each **s** and count each **s** proportional to its probability.

or

- Calculate $p(s_i \mid \mathbf{w}, \boldsymbol{\gamma}, \boldsymbol{\eta})$ for each i, and count each emission of w_i from s_i proportional to its probability.
- Calculate $p(s_i\ s_{i+1} \mid \mathbf{w}, \boldsymbol{\gamma}, \boldsymbol{\eta})$ for each i, and count each transition from s_i to s_{i+1} proportional to its probability.

E Step: Per-Word and Pairwise Posteriors

- Given **w** and current parameters, run forward and backward algorithms to obtain, for each i, the posteriors:
  - $p(s_i \mid \mathbf{w}, \boldsymbol{\gamma}, \boldsymbol{\eta})$
  - $p(s_i\ s_{i+1} \mid \mathbf{w}, \boldsymbol{\gamma}, \boldsymbol{\eta})$
- Think of these as *soft* or *fractional* counts of transition and emission events.

Soft Counts from Posteriors

$$
\begin{aligned}
\widehat{\text{freq}}(\sigma\sigma') &= \sum_i p(S_i = \sigma, S_{i+1} = \sigma' \mid \boldsymbol{w}, \boldsymbol{\gamma}, \boldsymbol{\eta}) \\
&= \sum_i f_i(\sigma) \times \gamma(\sigma' \mid \sigma) \times \eta(w_{i+1} \mid \sigma') \times b_i(\sigma') \\
\widehat{\text{freq}}\begin{pmatrix} \sigma \\ w \end{pmatrix} &= \sum_{i:w_i=w} p(S_i = \sigma \mid \boldsymbol{w}, \boldsymbol{\gamma}, \boldsymbol{\eta}) \\
&= \sum_{i:w_i=w} f_i(\sigma) \times b_i(\sigma) \\
\widehat{\text{freq}}(\sigma) &= \sum_i p(S_i = \sigma \mid \boldsymbol{w}, \boldsymbol{\gamma}, \boldsymbol{\eta}) \\
&= \sum_i p(S_i = \sigma \mid \boldsymbol{w}, \boldsymbol{\gamma}, \boldsymbol{\eta})
\end{aligned}
$$

Procedural View of EM

- Begin with an initial estimate of the parameters of the HMM ($\boldsymbol{\gamma}$ and $\boldsymbol{\eta}$).
- Iterate:
  - E step: calculate probability of each possible transition and each possible emission at each position.
  - M step: re-estimate parameters to maximize likelihood of "complete" data.

M Step: MLE by Relative Frequencies

- When we observed all the data, we used hard counts:

$$\hat{\gamma}(\sigma \mid \sigma') = \frac{\text{freq}(\sigma\sigma')}{\text{freq}(\sigma')}$$

$$\hat{\eta}(w \mid \sigma) = \frac{\text{freq}\begin{pmatrix}\sigma \\ w\end{pmatrix}}{\text{freq}(\sigma)}$$

- Now we do the same with soft counts:

$$\hat{\gamma}(\sigma \mid \sigma') = \frac{\widehat{\text{freq}}(\sigma'\sigma)}{\widehat{\text{freq}}(\sigma')}$$

$$\hat{\eta}(w \mid \sigma) = \frac{\widehat{\text{freq}}\begin{pmatrix}\sigma \\ w\end{pmatrix}}{\widehat{\text{freq}}(\sigma)}$$

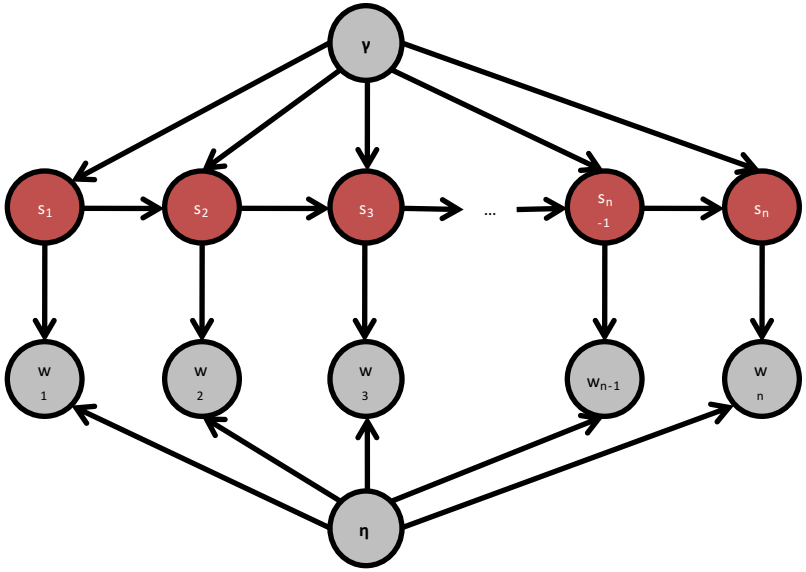
EM: Assurances

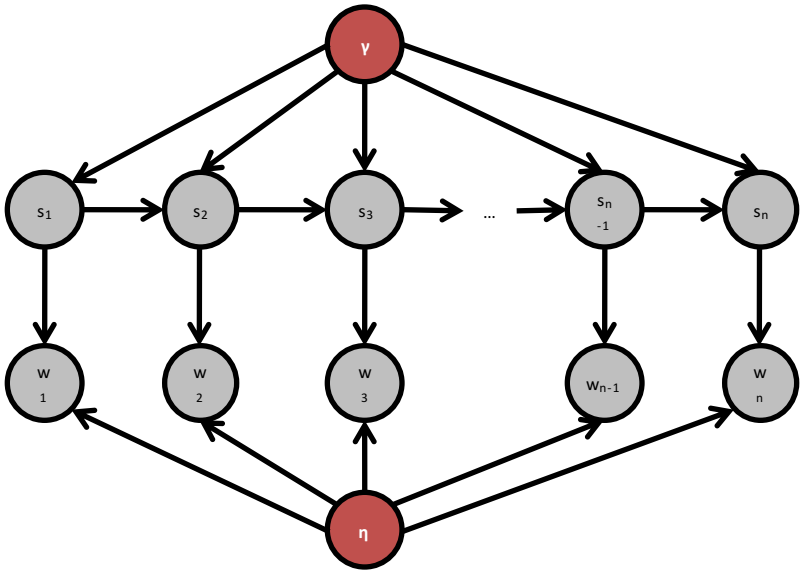
- Each iteration of EM will give us an estimate with a better likelihood than the last.
- Eventually we will converge to a *local* optimum (or saddle point).
  - We usually do not worry about saddle points.
  - Where you end up depends on where you start.

The Importance of Initialization

“Hard” EM

- Instead of using forward-backward to get fractional counts, we can use Viterbi to get hard counts.
- This equates to:





“Hard” EM

- Can be understood as coordinate ascent on this maximization problem:

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

- Some people prefer this.
  - Faster to converge (but to a different solution).
  - Viterbi algorithm instead of forward-backward.
  - Maybe more brittle.

Terminology

- Expectation-maximization (EM) is a very general technique, not just for HMMs.
  - Applicable to *any* generative model!
  - You may have seen it for mixtures of Gaussians or other kinds of clustering.
  - K-means clustering is a kind of hard EM.
- Sometimes the HMM version is called **Baum-Welch** training or **forward-backward** training.

HMMS AND
WEIGHTED FINITE-STATE MACHINES

Finite-State Machines

From formal language theory and theory of computation:

- Finite set of states.
- Set of allowed transitions between states.
- Nondeterministic walk among states.
- Each state (alternately, each transition) generates a symbol.

HMMs are Probabilistic FSAs

Go from “nondeterministic” to “probabilistic.”

- Put a probability distribution on the transitions out of each state.
- Put a probability distribution on the emissions from each state.

Powerful Generalization

- If the finite-state machine reads one sequence and “transcribes” it into another sequence, we have a **finite-state transducer**.
  - Sometimes: “read one tape and write one tape.”
- We can make these probabilistic as well, in a lot of different ways.
- Allows composition of stochastic relations, or chaining together of string-to-string transformations.
- Algorithms for inference and learning are similar to what we have seen.

Examples of Composed FSTs

- Speech recognition:
  Acoustic signal → pronounced phonemes → canonicalized words.
- Translation:
  Words in Czech → morphemes in Czech → morphemes in Slovak → words in Slovak

More Advanced Topics

- Feature-based parameterizations of HMMs and FSTs
  - Representation learning (spectral, neural, ...)
- Discriminative versions of HMMs (e.g., CRFs, seq2seq)
- Weakening independence assumptions (e.g., semi-HMMs)
- Method of moments estimation ("spectral")
- Generalizing HMMs and FSTs to probabilistic and weighted context-free grammars (and beyond) to model long-distance interactions and reordering.
- Bayesian inference and learning, including nonparametric priors (e.g., the "infinite" HMM)
- The relationship between WFSAs and some RNNs

Lecture Outline

- ✓ Markov models
- ✓ Hidden Markov models
- ✓ Viterbi algorithm
- ✓ Other inference algorithms for HMMs
- ✓ Learning algorithms for HMMs

Thanks!