Department of Computer Science
UNIVERSITY OF COLORADO **BOULDER**

Hidden Markov Models

Natural Language Processing: Jordan Boyd-Graber
University of Colorado Boulder

LECTURE 20

Adapted from material by Ray Mooney

Roadmap

- Classification: labeling one thing at a time
- Sometimes context matters
- **Sequence Labeling**: Classification over a string
- Hidden Markov Models: Generative sequence labeling algorithm

Sequence Labeling Tasks

- When has a credit card been compromised?
- What's the binding site of a protein?
- When are people sleeping (based on fitbits)?
- What is the part of speech of a word?

POS Tagging: Task Definition

- Annotate each word in a sentence with a part-of-speech marker.
- Lowest level of syntactic analysis.

John	saw	the	saw	and	decided	to	take	it	to	the	table
NNP	VBD	DT	NN	CC	VBD	TO	VB	PRP	IN	DT	NN

Tag Examples

- Noun (person, place or thing)
  - Singular (NN): dog, fork
  - Plural (NNS): dogs, forks
  - Proper (NNP, NNPS): John, Springfields
- Personal pronoun (PRP): I, you, he, she, it
- Wh-pronoun (WP): who, what
- Verb (actions and processes)
  - Base, infinitive (VB): eat
  - Past tense (VBD): ate
  - Gerund (VBG): eating
  - Past participle (VBN): eaten
  - Non 3rd person singular present tense (VBP): eat
  - 3rd person singular present tense: (VBZ): eats
  - Modal (MD): should, can
  - To (TO): to (to eat)

Ambiguity

"Like" can be a verb or a preposition

- I like/VBP candy.
- Time flies like/IN an arrow.

"Around" can be a preposition, particle, or adverb

- I bought it at the shop around/IN the corner.
- I never got around/RP to getting a car.
- A new Prius costs around/RB $25K.

How hard is it?

- Usually assume a separate initial tokenization process that separates and/or disambiguates punctuation, including detecting sentence boundaries.
- Degree of ambiguity in English (based on Brown corpus)
  - 11.5% of word types are ambiguous.
  - 40% of word tokens are ambiguous.
- Average POS tagging disagreement amongst expert human judges for the Penn treebank was 3.5%
- Based on correcting the output of an initial automated tagger, which was deemed to be more accurate than tagging from scratch.
- Baseline: Picking the most frequent tag for each specific word type gives about 90% accuracy 93.7% if use model for unknown words for Penn Treebank tagset.

What about classification / feature engineering?

- Just predict the most frequent class
- 0.38 accuracy
- Can get to around 60% accuracy by adding in dictionaries, prefix / suffix features

A more fundamental problem . . .

- Each classification is independent . . .
- This isn't right!
- If you have a noun, it's more likely to be preceeded by an adjective
- Determiners are followed by either a noun or an adjective
- Determiners don't follow each other

Approaches

- Rule-Based: Human crafted rules based on lexical and other linguistic knowledge.
- Learning-Based: Trained on human annotated corpora like the Penn Treebank.
    - Statistical models: Hidden Markov Model (HMM), Maximum Entropy Markov Model (MEMM), Conditional Random Field (CRF)
    - Rule learning: Transformation Based Learning (TBL)
- Generally, learning-based approaches have been found to be more effective overall, taking into account the total amount of human expertise and effort involved.

Approaches

- Rule-Based: Human crafted rules based on lexical and other linguistic knowledge.
- Learning-Based: Trained on human annotated corpora like the Penn Treebank.
  - Statistical models: Hidden Markov Model (HMM), Maximum Entropy Markov Model (MEMM), Conditional Random Field (CRF)
  - Rule learning: Transformation Based Learning (TBL)
- Generally, learning-based approaches have been found to be more effective overall, taking into account the total amount of human expertise and effort involved.

Outline

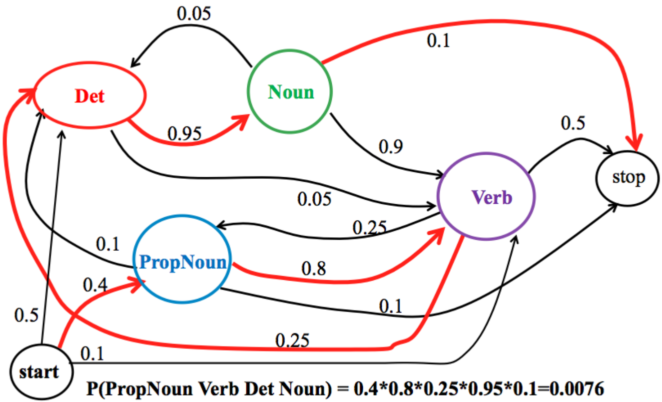
HMM Definition

- A finite state machine with probabilistic state transitions.
- Makes Markov assumption that next state only depends on the current state and independent of previous history.

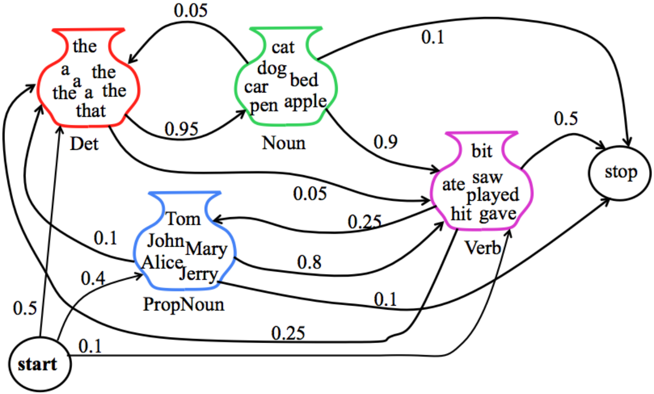
Generative Model

- Probabilistic generative model for sequences.
- Assume an underlying set of hidden (unobserved) states in which the model can be (e.g. parts of speech).
- Assume probabilistic transitions between states over time (e.g. transition from POS to another POS as sequence is generated).
- Assume a probabilistic generation of tokens from states (e.g. words generated for each POS).

Cartoon


0.05
0.1
Det
Noun
0.95
0.5
0.9
stop
0.05
Verb
0.25
0.1
PropNoun
0.8
0.4
0.1
0.5
0.25
0.1
start
P(PropNoun Verb Det Noun) = 0.4*0.8*0.25*0.95*0.1=0.0076

Cartoon


0.05
the
a a the
the a the
that
Det
cat
dog bed
car
pen apple
Noun
0.1
0.95
0.9
0.5
bit
ate saw
played
hit gave
Verb
stop
0.05
0.25
Tom
John Mary
Alice Jerry
PropNoun
0.1
0.8
0.4
0.1
0.5
0.25
0.1
start

Outline

HMM Definition

Assume K parts of speech, a lexicon size of V, a series of observations $\{x_1, \dots, x_N\}$, and a series of unobserved states $\{z_1, \dots, z_N\}$.

- π A distribution over start states (vector of length K): $\pi_i = p(z_1 = i)$
- θ Transition matrix (matrix of size K by K): $\theta_{i,j} = p(z_n = j | z_{n-1} = i)$
- β An emission matrix (matrix of size K by V): $\beta_{j,w} = p(x_n = w | z_n = j)$

HMM Definition

Assume K parts of speech, a lexicon size of V, a series of observations $\{x_1, \ldots, x_N\}$, and a series of unobserved states $\{z_1, \ldots, z_N\}$.

- π A distribution over start states (vector of length K):
  $\pi_i = p(z_1 = i)$
- θ Transition matrix (matrix of size K by K):
  $\theta_{i,j} = p(z_n = j | z_{n-1} = i)$
- β An emission matrix (matrix of size K by V):
  $\beta_{j,w} = p(x_n = w | z_n = j)$

Two problems: How do we move from data to a model? (Estimation) How do we move from a model and unlabled data to labeled data? (Inference)

Outline

Reminder: How do we estimate a probability?

- For a multinomial distribution (i.e. a discrete distribution, like over words):

$$\theta_i = \frac{n_i + \alpha_i}{\sum_k n_k + \alpha_k} \tag{1}$$

- α_i is called a smoothing factor, a pseudocount, etc.

Reminder: How do we estimate a probability?

- For a multinomial distribution (i.e. a discrete distribution, like over words):

$$\theta_i = \frac{n_i + \alpha_i}{\sum_k n_k + \alpha_k} \tag{1}$$

- α_i is called a smoothing factor, a pseudocount, etc.
- When $\alpha_i = 1$ for all i, it's called "Laplace smoothing" and corresponds to a uniform prior over all multinomial distributions.

Training Sentences

here	come	old	flattop
MOD	V	MOD	N

a	crowd	of	people	stopped	and	stared
DET	N	PREP	N	V	CONJ	V

gotta	get	you	into	my	life
V	V	PRO	PREP	PRO	V

and	I	love	her
CONJ	PRO	V	PRO

Training Sentences

x	here	come	old	flattop
	MOD	V	MOD	N

a	crowd	of	people	stopped	and	stared
DET	N	PREP	N	V	CONJ	V

gotta	get	you	into	my	life
V	V	PRO	PREP	PRO	V

and	I	love	her
CONJ	PRO	V	PRO

Training Sentences

x	here	come	old	flattop
z	MOD	V	MOD	N

a	crowd	of	people	stopped	and	stared
DET	N	PREP	N	V	CONJ	V

gotta	get	you	into	my	life
V	V	PRO	PREP	PRO	V

and	I	love	her
CONJ	PRO	V	PRO

Initial Probability π

POS	Frequency	Probability
MOD	1.1	0.234
DET	1.1	0.234
CONJ	1.1	0.234
N	0.1	0.021
PREP	0.1	0.021
PRO	0.1	0.021
V	1.1	0.234

Remember, we're taking MAP estimates, so we add 0.1 (arbitrarily chosen) to each of the counts before normalizing to create a probability distribution. This is easy; one sentence starts with an adjective, one with a determiner, one with a verb, and one with a conjunction.

Training Sentences

here	come	old	flattop
MOD	V	MOD	N

a	crowd	of	people	stopped	and	stared
DET	N	PREP	N	V	CONJ	V

gotta	get	you	into	my	life
V	V	PRO	PREP	PRO	N

and	I	love	her
CONJ	PRO	V	PRO

Training Sentences

here	come	old	flattop
MOD	V	MOD	N

a	crowd	of	people	stopped	and	stared
DET	N	PREP	N	V	CONJ	V

gotta	get	you	into	my	life
V	V	PRO	PREP	PRO	N

and	I	love	her
CONJ	PRO	V	PRO

Training Sentences

here	come	old	flattop
MOD	V	MOD	N

a	crowd	of	people	stopped	and	stared
DET	N	PREP	N	V	CONJ	V

gotta	get	you	into	my	life
V	V	PRO	PREP	PRO	N

and	I	love	her
CONJ	PRO	V	PRO

Transition Probability θ

- We can ignore the words; just look at the parts of speech. Let's compute one row, the row for verbs.
- We see the following transitions: V $\rightarrow$ MOD, V $\rightarrow$ CONJ, V $\rightarrow$ V, V $\rightarrow$ PRO, and V $\rightarrow$ PRO

POS	Frequency	Probability
MOD	1.1	0.193
DET	0.1	0.018
CONJ	1.1	0.193
N	0.1	0.018
PREP	0.1	0.018
PRO	2.1	0.368
V	1.1	0.193

- And do the same for each part of speech ...

Training Sentences

here	come	old	flattop
MOD	V	MOD	N

a	crowd	of	people	stopped	and	stared
DET	N	PREP	N	V	CONJ	V

gotta	get	you	into	my	life
V	V	PRO	PREP	PRO	N

and	I	love	her
CONJ	PRO	V	PRO

Training Sentences

here	come	old	flattop
MOD	V	MOD	N

a	crowd	of	people	stopped	and	stared
DET	N	PREP	N	V	CONJ	V

gotta	get	you	into	my	life
V	V	PRO	PREP	PRO	N

and	I	love	her
CONJ	PRO	V	PRO

Emission Probability β

Let's look at verbs . . .

Word	a	and	come	crowd	flattop
Frequency	0.1	0.1	1.1	0.1	0.1
Probability	0.0125	0.0125	0.1375	0.0125	0.0125

Word	get	gotta	her	here	i
Frequency	1.1	1.1	0.1	0.1	0.1
Probability	0.1375	0.1375	0.0125	0.0125	0.0125

Word	into	it	life	love	my
Frequency	0.1	0.1	0.1	1.1	0.1
Probability	0.0125	0.0125	0.0125	0.1375	0.0125

Word	of	old	people	stared	stopped
Frequency	0.1	0.1	0.1	1.1	1.1
Probability	0.0125	0.0125	0.0125	0.1375	0.1375

Next time . . .

- Viterbi algorithm: dynamic algorithm discovering the most likely POS sequence given a sentence
- EM algorithm: what if we don't have labeled data?

Outline

Viterbi Algorithm

- Given an unobserved sequence of length L, $\{x_1, \ldots, x_L\}$, we want to find a sequence $\{z_1 \ldots z_L\}$ with the highest probability.

Viterbi Algorithm

- Given an unobserved sequence of length L, $\{x_1, \dots, x_L\}$, we want to find a sequence $\{z_1 \dots z_L\}$ with the highest probability.
- It's impossible to compute K^L possibilities.
- So, we use dynamic programming to compute most likely tags for each token subsequence from 0 to t that ends in state k.
- Memoization: fill a table of solutions of sub-problems
- Solve larger problems by composing sub-solutions
- Base case:

$$\delta_1(k) = \pi_k \beta_{k,x_i} \tag{2}$$

- Recursion:

$$\delta_n(k) = \max_j \left(\delta_{n-1}(j)\theta_{j,k}\right)\beta_{k,x_n} \tag{3}$$

Viterbi Algorithm

- Given an unobserved sequence of length L, $\{x_1, \dots, x_L\}$, we want to find a sequence $\{z_1 \dots z_L\}$ with the highest probability.
- It's impossible to compute K^L possibilities.
- So, we use dynamic programming to compute most likely tags for each token subsequence from 0 to t that ends in state k.
- Memoization: fill a table of solutions of sub-problems
- Solve larger problems by composing sub-solutions
- Base case:

$$\delta_1(k) = \pi_k \beta_{k,x_i} \tag{2}$$

- Recursion:

$$\delta_n(k) = \max_{j} \left(\delta_{n-1}(j)\theta_{j,k}\right)\beta_{k,x_n} \tag{3}$$

Viterbi Algorithm

- Given an unobserved sequence of length L, $\{x_1, \dots, x_L\}$, we want to find a sequence $\{z_1 \dots z_L\}$ with the highest probability.
- It's impossible to compute K^L possibilities.
- So, we use dynamic programming to compute most likely tags for each token subsequence from 0 to t that ends in state k.
- Memoization: fill a table of solutions of sub-problems
- Solve larger problems by composing sub-solutions
- Base case:

$$\delta_1(k) = \pi_k \beta_{k,x_i} \tag{2}$$

- Recursion:

$$\delta_n(k) = \max_j \left(\delta_{n-1}(j)\theta_{j,k}\right)\beta_{k,x_n} \tag{3}$$

Viterbi Algorithm

- Given an unobserved sequence of length L, $\{x_1, \ldots, x_L\}$, we want to find a sequence $\{z_1 \ldots z_L\}$ with the highest probability.
- It's impossible to compute K^L possibilities.
- So, we use dynamic programming to compute most likely tags for each token subsequence from 0 to t that ends in state k.
- Memoization: fill a table of solutions of sub-problems
- Solve larger problems by composing sub-solutions
- Base case:

$$\delta_1(k) = \pi_k \beta_{k,x_i} \tag{2}$$

- Recursion:

$$\delta_n(k) = \max_j \left(\delta_{n-1}(j)\theta_{j,k}\right)\beta_{k,x_n} \tag{3}$$

- The complexity of this is now $K^2 L$.
- In class: example that shows why you need all $O(KL)$ table cells (garden pathing)
- But just computing the max isn't enough. We also have to remember where we came from. (Breadcrumbs from best previous state.)

$$\Psi_n = \text{argmax}_j \delta_{n-1}(j)\theta_{j,k} \tag{4}$$

- The complexity of this is now K^2L.
- In class: example that shows why you need all $O(KL)$ table cells (garden pathing)
- But just computing the max isn't enough. We also have to remember where we came from. (Breadcrumbs from best previous state.)

$$\Psi_n = \text{argmax}_j \delta_{n-1}(j)\theta_{j,k} \tag{4}$$

- Let's do that for the sentence "come and get it"

Outline

POS	π_k	β_{k,x_1}	$\log \delta_1(k)$
MOD	0.234	0.024	-5.18
DET	0.234	0.032	-4.89
CONJ	0.234	0.024	-5.18
N	0.021	0.016	-7.99
PREP	0.021	0.024	-7.59
PRO	0.021	0.016	-7.99
V	0.234	0.121	-3.56

come and get it

Why logarithms?

1. More interpretable than a float with lots of zeros.
2. Underflow is less of an issue
3. Addition is cheaper than multiplication

$$log(ab) = log(a) + log(b) \tag{5}$$

POS	$\log \delta_1(j)$		$\log \delta_2(\text{CONJ})$
MOD	-5.18		
DET	-4.89		
CONJ	-5.18		
N	-7.99		
PREP	-7.59		
PRO	-7.99		
V	-3.56		

come **and** get it

POS	$\log \delta_1(j)$		$\log \delta_2(\text{CONJ})$
MOD	-5.18		
DET	-4.89		
CONJ	-5.18		???
N	-7.99		
PREP	-7.59		
PRO	-7.99		
V	-3.56		

come **and** get it

POS	$\log \delta_1(j)$	$\log \delta_1(j)\theta_{j,\text{CONJ}}$	$\log \delta_2(\text{CONJ})$
MOD	-5.18		
DET	-4.89		
CONJ	-5.18		???
N	-7.99		
PREP	-7.59		
PRO	-7.99		
V	-3.56		

come **and** get it

POS	$\log \delta_1(j)$	$\log \delta_1(j)\theta_{j,\text{CONJ}}$	$\log \delta_2(\text{CONJ})$
MOD	-5.18		
DET	-4.89		
CONJ	-5.18		???
N	-7.99		
PREP	-7.59		
PRO	-7.99		
V	-3.56		

come **and** get it

$$\log \left(\delta_0(\text{V})\theta_{\text{V, CONJ}}\right) = \log \delta_0(k) + \log \theta_{\text{V, CONJ}} = -3.56 + -1.65$$

POS	$\log \delta_1(j)$	$\log \delta_1(j)\theta_{j,\text{CONJ}}$	$\log \delta_2(\text{CONJ})$
MOD	-5.18		
DET	-4.89		
CONJ	-5.18		???
N	-7.99		
PREP	-7.59		
PRO	-7.99		
V	-3.56	-5.21	

come **and** get it

POS	$\log \delta_1(j)$	$\log \delta_1(j)\theta_{j,\text{CONJ}}$	$\log \delta_2(\text{CONJ})$
MOD	-5.18		
DET	-4.89		
CONJ	-5.18		???
N	-7.99	≤ -7.99	
PREP	-7.59	≤ -7.59	
PRO	-7.99	≤ -7.99	
V	-3.56	-5.21	

come **and** get it

POS	$\log \delta_1(j)$	$\log \delta_1(j)\theta_{j,\text{CONJ}}$	$\log \delta_2(\text{CONJ})$
MOD	-5.18	-8.48	
DET	-4.89	-7.72	
CONJ	-5.18	-8.47	???
N	-7.99	≤ -7.99	
PREP	-7.59	≤ -7.59	
PRO	-7.99	≤ -7.99	
V	-3.56	-5.21	

come **and** get it

POS	$\log \delta_1(j)$	$\log \delta_1(j)\theta_{j,\text{CONJ}}$	$\log \delta_2(\text{CONJ})$
MOD	-5.18	-8.48	
DET	-4.89	-7.72	
CONJ	-5.18	-8.47	???
N	-7.99	≤ -7.99	
PREP	-7.59	≤ -7.59	
PRO	-7.99	≤ -7.99	
V	-3.56	-5.21	

come **and** get it

POS	$\log \delta_1(j)$	$\log \delta_1(j)\theta_{j,\text{CONJ}}$	$\log \delta_2(\text{CONJ})$
MOD	-5.18	-8.48	
DET	-4.89	-7.72	
CONJ	-5.18	-8.47	
N	-7.99	≤ -7.99	
PREP	-7.59	≤ -7.59	
PRO	-7.99	≤ -7.99	
V	-3.56	-5.21	

come **and** get it

$$\log \delta_1(k) = -5.21 - \log \beta_{\text{CONJ, and}} =$$

POS	$\log \delta_1(j)$	$\log \delta_1(j)\theta_{j,\text{CONJ}}$	$\log \delta_2(\text{CONJ})$
MOD	-5.18	-8.48	
DET	-4.89	-7.72	
CONJ	-5.18	-8.47	
N	-7.99	≤ -7.99	
PREP	-7.59	≤ -7.59	
PRO	-7.99	≤ -7.99	
V	-3.56	-5.21	

come **and** get it

$$\log \delta_1(k) = -5.21 - \log \beta_{\text{CONJ, and}} = -5.21 - 0.64$$

POS	$\log \delta_1(j)$	$\log \delta_1(j)\theta_{j,\text{CONJ}}$	$\log \delta_2(\text{CONJ})$
MOD	-5.18	-8.48	
DET	-4.89	-7.72	
CONJ	-5.18	-8.47	-6.02
N	-7.99	≤ -7.99	
PREP	-7.59	≤ -7.59	
PRO	-7.99	≤ -7.99	
V	-3.56	-5.21	

come **and** get it

POS	$\delta_1(k)$	$\delta_2(k)$	b_2	$\delta_3(k)$	b_3	$\delta_4(k)$	b_4
MOD	-5.18						
DET	-4.89						
CONJ	-5.18	-6.02	V				
N	-7.99						
PREP	-7.59						
PRO	-7.99						
V	-3.56						
WORD	come	and		get		it	

POS	$\delta_1(k)$	$\delta_2(k)$	b_2	$\delta_3(k)$	b_3	$\delta_4(k)$	b_4
MOD	-5.18	-0.00	X				
DET	-4.89	-0.00	X				
CONJ	-5.18	-6.02	V				
N	-7.99	-0.00	X				
PREP	-7.59	-0.00	X				
PRO	-7.99	-0.00	X				
V	-3.56	-0.00	X				
WORD	come	and		get		it	

POS	$\delta_1(k)$	$\delta_2(k)$	b_2	$\delta_3(k)$	b_3	$\delta_4(k)$	b_4
MOD	-5.18	-0.00	X	-0.00	X		
DET	-4.89	-0.00	X	-0.00	X		
CONJ	-5.18	-6.02	V	-0.00	X		
N	-7.99	-0.00	X	-0.00	X		
PREP	-7.59	-0.00	X	-0.00	X		
PRO	-7.99	-0.00	X	-0.00	X		
V	-3.56	-0.00	X	-9.03	CONJ		
WORD	come	and		get		it	

POS	$\delta_1(k)$	$\delta_2(k)$	b_2	$\delta_3(k)$	b_3	$\delta_4(k)$	b_4
MOD	-5.18	-0.00	X	-0.00	X	-0.00	X
DET	-4.89	-0.00	X	-0.00	X	-0.00	X
CONJ	-5.18	-6.02	V	-0.00	X	-0.00	X
N	-7.99	-0.00	X	-0.00	X	-0.00	X
PREP	-7.59	-0.00	X	-0.00	X	-0.00	X
PRO	-7.99	-0.00	X	-0.00	X	-14.6	V
V	-3.56	-0.00	X	-9.03	CONJ	-0.00	X
WORD	come	and		get		it	

Outline

What if you don't have training data?

- You can still learn a HMM
- Using a general technique called expectation maximization

What if you don't have training data?

- You can still learn a HMM
- Using a general technique called expectation maximization
  - Take a guess at the parameters
  - Figure out latent variables
  - Find the parameters that best explain the latent variables
  - Repeat

em for hmm

Model Parameters

We need to start with model parameters

em for hmm

Model Parameters

π, β, θ

We can initialize these any way we want

em for hmm

Model Parameters

π, β, θ

E step

em for hmm

Model Parameters

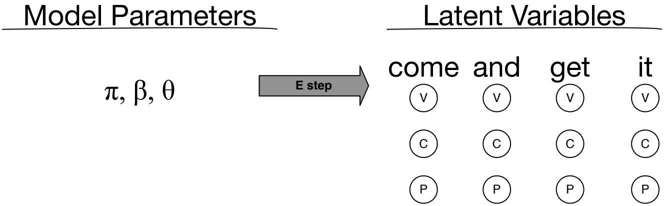
π, β, θ $\xrightarrow{\text{E step}}$

Latent Variables

come and get it

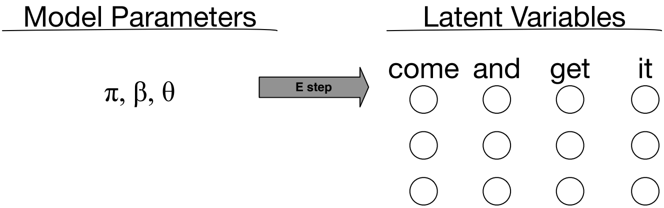
We compute the E-step based on our data

em for hmm



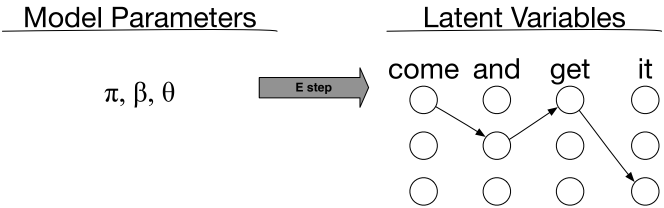
Each word in our dataset could take any part of speech

em for hmm



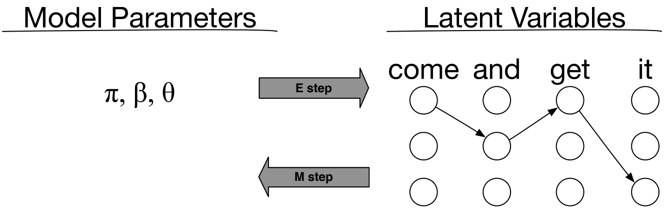
But we don't know which state was used for each word

em for hmm



Determine the probability of being in each latent state using Forward / Backward

em for hmm

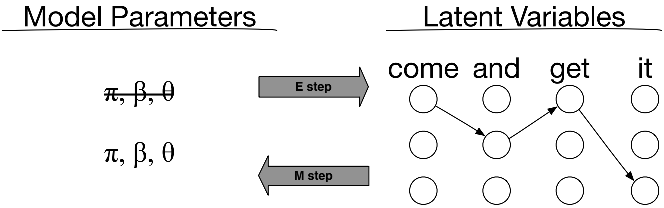


Calculate new parameters:

$$\theta_i = \frac{n_i + \alpha_i}{\sum_k \mathbb{E}_p[n_k] + \alpha_k} \quad (6)$$

Where the expected counts are from the lattice

em for hmm



Replace old parameters (and start over)

Hard vs. Full EM

Hard EM

Train only on the most likely sentence (Viterbi)

- Faster: E-step is faster
- Faster: Fewer iterations

Full EM

Compute probability of all possible sequences

- More accurate: Doesn't get stuck in local optima as easily

Recap

- Generative model for sequence labeling
- With example of part of speech tagging
- Next time: discriminative sequence labeling