Applications of HMMs

Any sequence tagging problem

- Named Entity Recognition

  `Alan Spoon , recently named Newsweek president ,`
  `said Newsweek ad rates would increase in January .`

- Morpheme Boundary Detection

  `Signs of a slowing economy are increasing`

- Simple Machine Translation (no reordering)

  `das blaue Haus`
  `the blue  house`

- Speech Recognition

  `testing one`

Generative Story for HMMs

- To create observation sequence $o_1 o_2 \ldots o_n$
  - For each time step i, generate a latent state s_i conditioned on s_{i-1}

    State Transition Model

  - For each time step i, generate an observation symbol o_i conditioned on s_i

    Channel Model

HMM Decoding

- Given observation, find best sequence of latent states

State Transition Model

$$\underset{\text{state sequences s}}{\operatorname{argmax}} \; P(s \mid o) = \underset{\text{state sequences s}}{\operatorname{argmax}} \; P(s)P(o \mid s)$$

Channel Model

  - What is the size of the search space (number of state sequences) for an observation of length n?

Dynamic Programming!

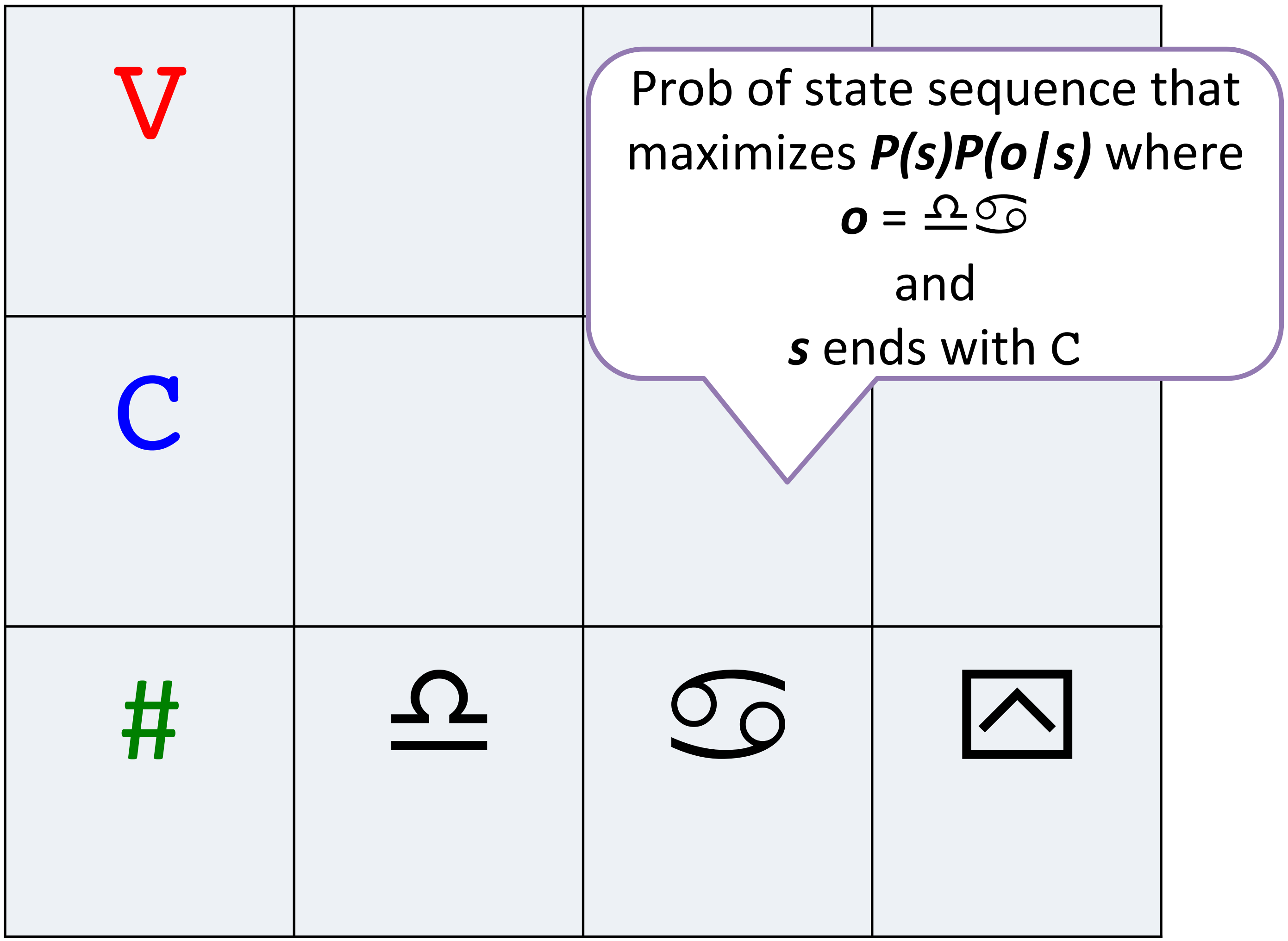
HMM Decoding Example

- I see this word from some language

♎♋⍓

- Which symbols are vowels and which are consonants?
  - Given: state (consonant-vowel) transition probabilities, channel "emission" probabilities

Viterbi Algorithm



Given these probabilities:

P(C|#)=0.7

P(C|C)=0.4

P(V|V)=0.1

P(♎|C)=0.08

P(♎|V)=0.01

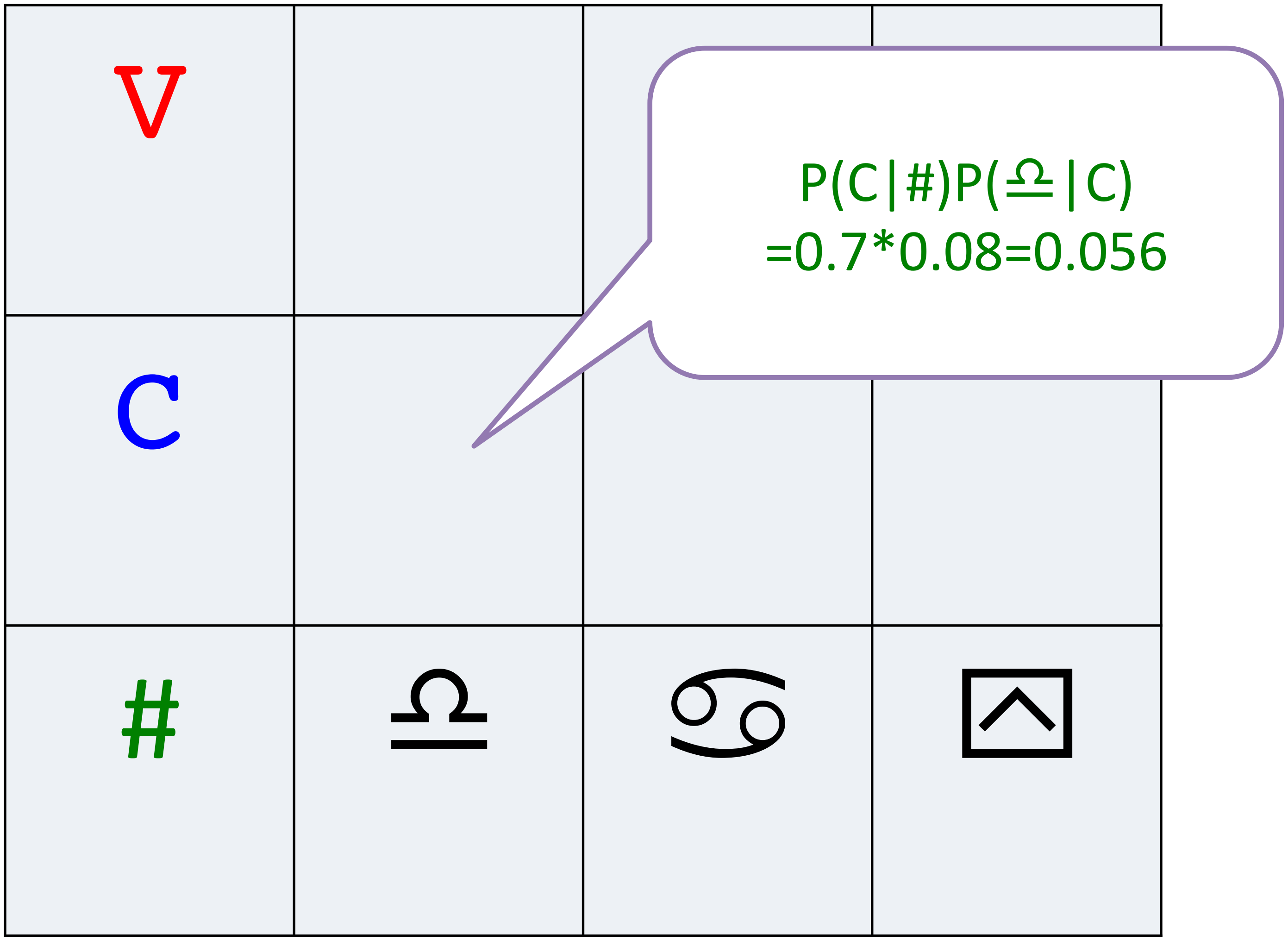
P(♋|C)=0.02

P(♋|V)=0.14

P(⍓|C)=0.07

P(⍓|V)=0.20

Viterbi Algorithm



Given these probabilities:

P(C|#)=0.7

P(C|C)=0.4

P(V|V)=0.1

P(♎|C)=0.08

P(♎|V)=0.01

P(♋|C)=0.02

P(♋|V)=0.14

P(⍓|C)=0.07

P(⍓|V)=0.20

Viterbi Algorithm

V			
C	0.056		
#	♎	♋	⍓

P(V|#)P(♎|V)
=0.3*0.01=0.003

Given these probabilities:
P(C|#)=0.7
P(C|C)=0.4
P(V|V)=0.1

P(♎|C)=0.08
P(♎|V)=0.01
P(♋|C)=0.02
P(♋|V)=0.14
P(⍓|C)=0.07
P(⍓|V)=0.20

Viterbi Algorithm

V	0.003		
C	0.056		
#	♎	♋	⍓

0.056*P(C|C)P(♋|C)
=0.056*0.4*0.02=0.000448
0.003*P(C|V)*P(♋|C)
=0.003*0.9*0.02=0.000054

Given these probabilities:
P(C|#)=0.7
P(C|C)=0.4
P(V|V)=0.1

P(♎|C)=0.08
P(♎|V)=0.01
P(♋|C)=0.02
P(♋|V)=0.14
P(⍓|C)=0.07
P(⍓|V)=0.20

Viterbi Algorithm

V	0.003		
C	0.056		
#	♎	♋	⍓

0.056*P(V|C)P(♋|V)
=0.056*0.6*0.14=0.004704
0.003*P(V|V)*P(♋|V)
=0.003*0.1*0.14=0.000042

Given these probabilities:

P(C|#)=0.7
P(C|C)=0.4
P(V|V)=0.1

P(♎|C)=0.08
P(♎|V)=0.01
P(♋|C)=0.02
P(♋|V)=0.14
P(⍓|C)=0.07
P(⍓|V)=0.20

Viterbi Algorithm

V	0.003	0.004704	
C	0.056	0.000448	
#	♎	♋	

0.000448*P(C|C)P(⍓|C)
=0.000448*0.4*0.07=0.0000125
0.004704*P(C|V)*P(⍓|C)
=0.004704*0.9*0.07=0.000296

Given these probabilities:
P(C|#)=0.7
P(C|C)=0.4
P(V|V)=0.1

P(♎|C)=0.08
P(♎|V)=0.01
P(♋|C)=0.02
P(♋|V)=0.14
P(⍓|C)=0.07
P(⍓|V)=0.20

Viterbi Algorithm

V	0.003	0.004704	
C	0.056	0.000448	0 296
#			

0.000448*P(V|C)P(⍓|V)
=0.000448*0.6*0.20=0.00005376
0.004704*P(V|V)*P(⍓|V)
=0.004704*0.1*0.20=0.0000941

Given these probabilities:

P(C|#)=0.7
P(C|C)=0.4
P(V|V)=0.1

P(♎|C)=0.08
P(♎|V)=0.01
P(♋|C)=0.02
P(♋|V)=0.14
P(⍓|C)=0.07
P(⍓|V)=0.20

Viterbi Algorithm

V	0.003	0.004704	0.000094
C	0.056	0.000448	0.000296
#	♎	♋	⍓

Given these probabilities:

P(C|#)=0.7

P(C|C)=0.4

P(V|V)=0.1

P(♎|C)=0.08

P(♎|V)=0.01

P(♋|C)=0.02

P(♋|V)=0.14

P(⍓|C)=0.07

P(⍓|V)=0.20

Viterbi Algorithm

V	0.003	0.004704	0.000094
C	0.056	0.000448	0.000296
#	♎	♋	⍓

Take the highest probability on the last column and trace back the pointers

Viterbi Algorithm

V	0.003	0.004704	0.000094
C	0.056	0.000448	0.000296
#	♎	♋	⍓

Take the highest probability on the last column and trace back the pointers

C
⍓

Viterbi Algorithm

V	0.003	0.004704	0.000094
C	0.056	0.000448	0.000296
#	♎	♋	⍓

Take the highest probability on the last column and trace back the pointers

V C

♋ ⍓

Viterbi Algorithm

V	0.003	0.004704	0.000094
C	0.056	0.000448	0.000296
#	♎	♋	⍓

Take the highest probability on the last column and trace back the pointers

C V C

♎ ♋ ⍓

Viterbi Algorithm

- Complexity?

  For sequence of length T and HMM with n states

  - Runtime is $O(n^2T)$: n x T cells, n computations per cell
  - Space is $O(nT)$

- Why does it work?

  That is, how does it find

$$\underset{\text{state sequences s}}{\text{argmax}} \; P(s \mid o) = \underset{\text{state sequences s}}{\text{argmax}} \; P(s)P(o \mid s)$$

  without enumerating all exponentially many state sequences?

  - Best state sequence for first few time steps will be part of eventual best state sequence. Only change comes from transition probability from time t to $t+1$

Convince yourself of this.

HMMs: Three Canonical Problems

- Finding the best sequence of states for an observation
  - Done: Viterbi Algorithm.
- Finding the total probability of an observation
  - Coming up on Thursday: Forward Algorithm
- Estimating the transition and emission probabilities from a corpus
  - Coming up next week: Baum-Welch

Why do we need to find the best state sequence?

- Part of speech tagging: resolving ambiguity

Noun	Verb or Noun?	Det	Noun
I	shot	an	elephant

- Predicting pronunciations of letters

K or S?	AA or AE?	R or sil?	S or Z?
c	a	r	s

Questions about Viterbi

- Difference between Markov Models and HMMs?
- What is the generative story for HMMs
- Where does $P(s)P(o|s)$ come from?
- What is end result produced by Viterbi?
- Where do the probabilities come from?
- Is there an algorithm-model relationship analogous to Viterbi-HMM?
- What does the Viterbi table really mean?

Why do we need to find the total probability of observation?

- Can use HMMs as a language model, instead of n-grams
- Recall: Language models assign probabilities to strings
  - Text Classification
  - Random Generation
  - Machine Translation
  - Speech Recognition

HMM Observation Probability

- Compute total probability of observation as generated by the HMM

State Transition Model

$$P(o) = \sum_{\text{state sequences } s} P(s,o) = \sum_{\text{state sequences } s} P(s)P(o \mid s)$$

Channel Model

HMM Best State Sequence

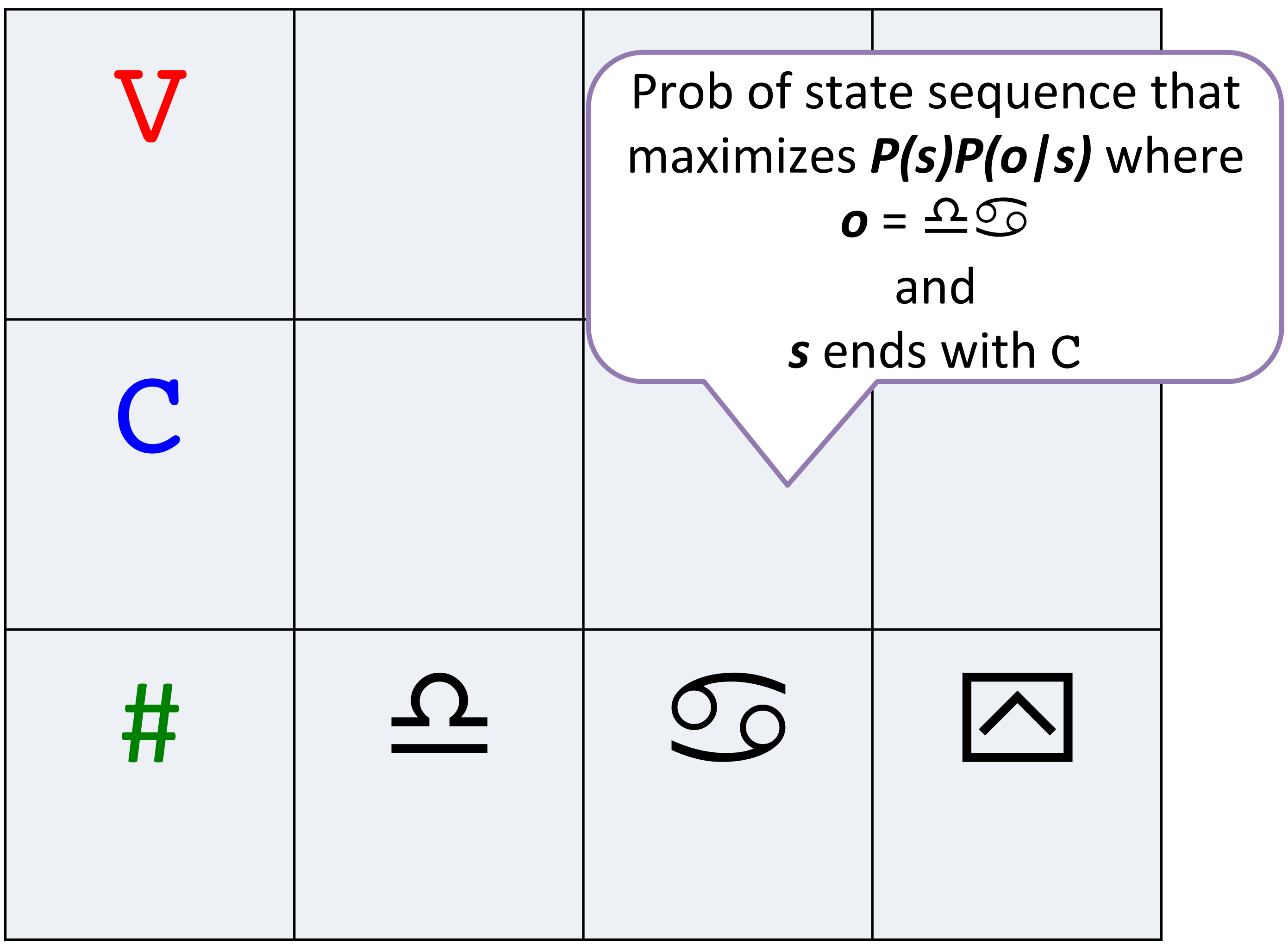
- Compare previous slide to this formula for the best state sequence for an observation (last class)

State Transition Model

$$\underset{\text{state sequences } s}{\operatorname{argmax}} P(s \mid o) = \underset{\text{state sequences } s}{\operatorname{argmax}} P(s)P(o \mid s)$$

Channel Model

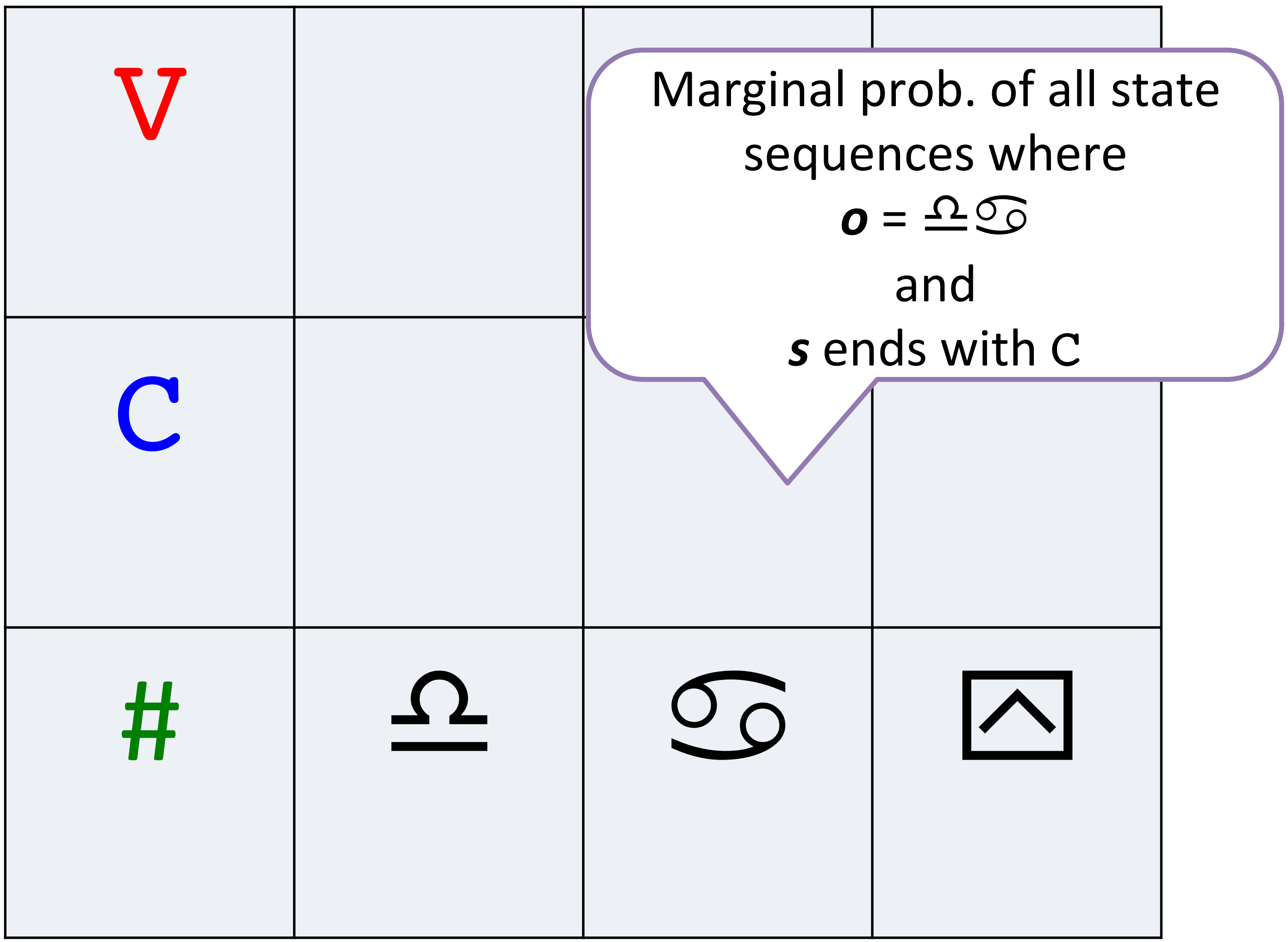
Recall: Viterbi Algorithm



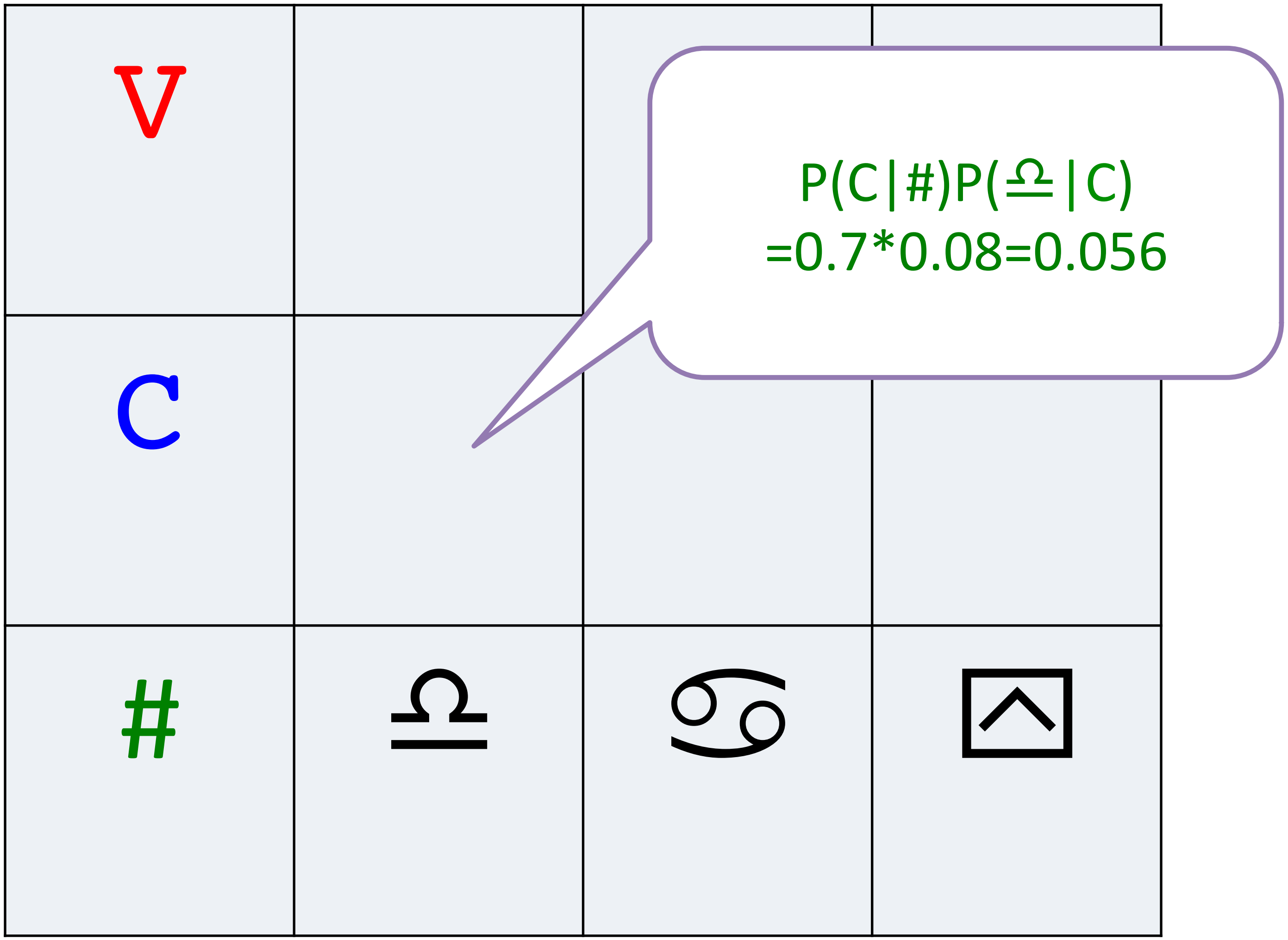
What small modification can we make to Viterbi?

Change what each cell represents…

Total Probability of Observation



Forward Algorithm



Given these probabilities:

P(C|#)=0.7

P(C|C)=0.4

P(V|V)=0.1

P(♎|C)=0.08

P(♎|V)=0.01

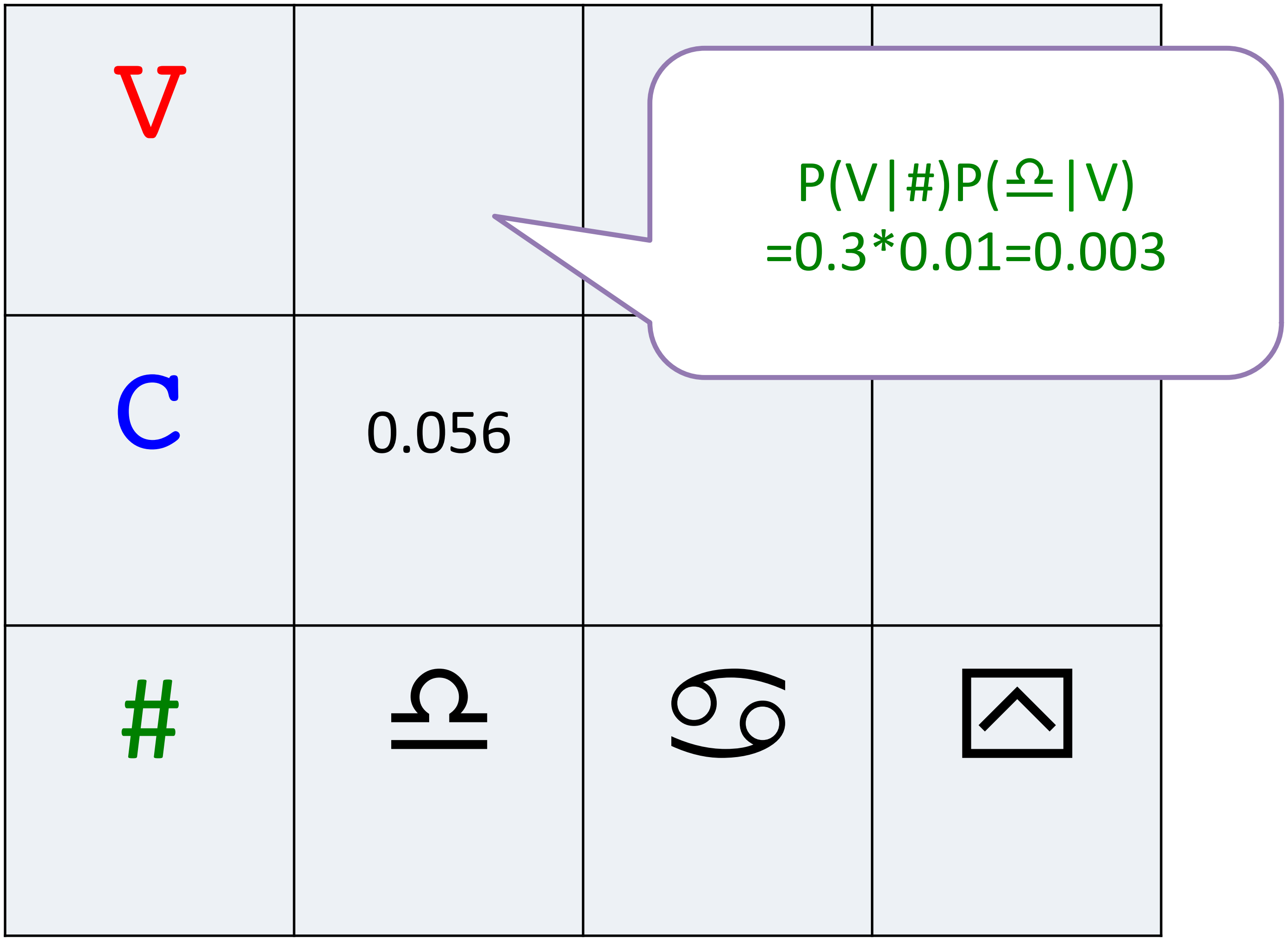
P(♋|C)=0.02

P(♋|V)=0.14

P(⍓|C)=0.07

P(⍓|V)=0.20

Forward Algorithm



Given these probabilities:

P(C|#)=0.7

P(C|C)=0.4

P(V|V)=0.1

P(♎|C)=0.08

P(♎|V)=0.01

P(♋|C)=0.02

P(♋|V)=0.14

P(⍓|C)=0.07

P(⍓|V)=0.20

Forward Algorithm

V	0.003		
C	0.056		
#	♎	♋	⍓

0.056*P(C|C)P(♋|C)
=0.056*0.4*0.02=0.000448
0.003*P(C|V)*P(♋|C)
=0.003*0.9*0.02=0.000054

Given these probabilities:
P(C|#)=0.7
P(C|C)=0.4
P(V|V)=0.1

P(♎|C)=0.08
P(♎|V)=0.01
P(♋|C)=0.02
P(♋|V)=0.14
P(⍓|C)=0.07
P(⍓|V)=0.20

Forw

0.056*P(V|C)P(♋|V)
=0.056*0.6*0.14=0.004704
0.003*P(V|V)*P(♋|V)
=0.003*0.1*0.14=0.000042

V	0.003		
C	0.056	0.000502	
#	♎	♋	⍓

Given these probabilities:

P(C|#)=0.7
P(C|C)=0.4
P(V|V)=0.1

P(♎|C)=0.08
P(♎|V)=0.01
P(♋|C)=0.02
P(♋|V)=0.14
P(⍓|C)=0.07
P(⍓|V)=0.20

For

V	0.003	0.004746	
C	0.056	0.000502	
#	♎	♋	⍓

0.000502*P(C|C)P(⍓|C)
=0.000502*0.4*0.07=0.000014

0.004746*P(C|V)*P(⍓|C)
=0.004746*0.9*0.07=0.000299

n these probabilities:

P(C|#)=0.7
P(C|C)=0.4
P(V|V)=0.1

P(♎|C)=0.08
P(♎|V)=0.01
P(♋|C)=0.02
P(♋|V)=0.14
P(⍓|C)=0.07
P(⍓|V)=0.20

For

0.000502*P(V|C)P(⍓|V)
=0.000502*0.6*0.2=0.000060

0.004746*P(V|V)*P(⍓|V)
=0.004746*0.1*0.2=0.000095

V	0.003	0.004746	
C	0.056	0.000502	0.000313
#	♎	♋	⍓

n these probabilities:

P(C|#)=0.7
P(C|C)=0.4
P(V|V)=0.1

P(♎|C)=0.08
P(♎|V)=0.01
P(♋|C)=0.02
P(♋|V)=0.14
P(⍓|C)=0.07
P(⍓|V)=0.20

Forward Algorithm

V	0.003	0.004746	0.000016
C	0.056	0.000502	0.000313
#	♎	♋	⍓

Given these probabilities:

P(C|#)=0.7
P(C|C)=0.4
P(V|V)=0.1

P(♎|C)=0.08
P(♎|V)=0.01
P(♋|C)=0.02
P(♋|V)=0.14
P(⍓|C)=0.07
P(⍓|V)=0.20

Forward Algorithm

V	0.003	0.004746	0.000016
C	0.056	0.000502	0.000313
#	♎	♋	⍓

Total probability of ♎♋⍓ under this model:

0.000016
+
0.000313
= 0.000329

Where do the parameters come from?

- Who gives these to us?
- Ideally, we want to estimate them from data
- Remember the maximum likelihood principle?
  - Find parameters that maximize probability of data, compared to all other parameter values

Maximum Likelihood Estimation

- HMM parameters:
  transition & emission probabilities
- How to find MLE parameters?
  - Suppose the data includes annotations of best hidden states for each time step

CVCVC VC V CCVVC CVC

♦□♎♋⍓ ♓⬧ ♋ ♑❒♏♋♦ ♎♋⍓

Easy: Just take relative frequencies!
(# times C→C, # times C→♎, etc. and normalize.)

Maximum Likelihood Estimation

- HMM parameters:
  transition & emission probabilities
- How to find MLE parameters?
  - What if the data does *not* contain annotations of best hidden states (more realistic)?

⧫□♎♋⍓ ♓⬧ ♋ ♑❒♏♋⧫ ♎♋⍓

Maximum Likelihood Estimation

- HMM parameters:
  transition & emission probabilities
- How to find MLE parameters?
  - What if the data does *not* contain annotations of best hidden states (more realistic)?
  - Can we get relative counts of state transitions and emissions then?

⧫□♎♋⍓ ♓⬧ ♋ ♑❒♏♋⧫ ♎♋⍓

Yes!

Unsupervised Learning

- Learn structure from data without any annotations about the structure
  - Eg: learn part-of-speech state transitions and emissions, given only the observed words

- But why?
  1. Isn't this how *we* learn?
  2. Getting annotations is difficult, expensive, sometimes impossible


Sometimes, I have some unlabeled data, and I want to put labels on it.
UNSUPERVISED LEARNING, AS SHE IS IMPLEMENTED.
So I write down a generative model, and then tell the data to find parameters that explain the data to me. And if I am not satisfied with the likelihood of this explanation, I tell the data to do it again until I am.
Wow, that sucks for the data.
I know, right? It's not the data's fault that I was too lazy to label it, right?
It seems like there are some deeper issues. Sometimes, most of the variation in the data comes from phenomena that are irrelevant to your desired labeling scheme. For example, the data might use its parameters to explain its semantics, when all you care about is its syntactic properties.
Why do we have to put labels on our data at all? Can't we just appreciate our data for who it is, and recognize that each datum is a unqiue and precious snowflake? Guys, all I'm saying, is maybe with a little supervision, our data can grow up to be whatever it wants to be!

What if…

- … we had a corpus of observation annotated with the state sequences?
  - Can estimate the MLE parameters by counting

- … we had the HMM parameters
  - Can find the best state sequences for the observations in the corpus with Viterbi

Expectation Maximization

1. Guess parameters of model
2. Tag observations with best state sequence
3. Re-estimate model parameters by counting
4. Repeat: go to step 2.

***Locking* parameters:**
If we like some of the original model parameters, can ignore the data and just not change those parameters.

Soft Expectation Maximization

1. Guess parameters of model
2. Tag observations with best state sequence
3. Re-estimate model parameters by counting
4. Repeat: go to step 2.

Bad guess can corner model into bad local maximum

Instead of getting counts from the best state sequence, get **expected counts** (weighted average) from all possible state sequences with their probabilities

Building block: Backward Algorithm

- Forward algorithm computes total probability of observation going left to right
- Backward algorithm computes total probability of observation going right to left

Backward Algorithm

Given these probabilities:

P(C|#)=0.7

P(C|C)=0.4

P(V|V)=0.1

P(♎|C)=0.08

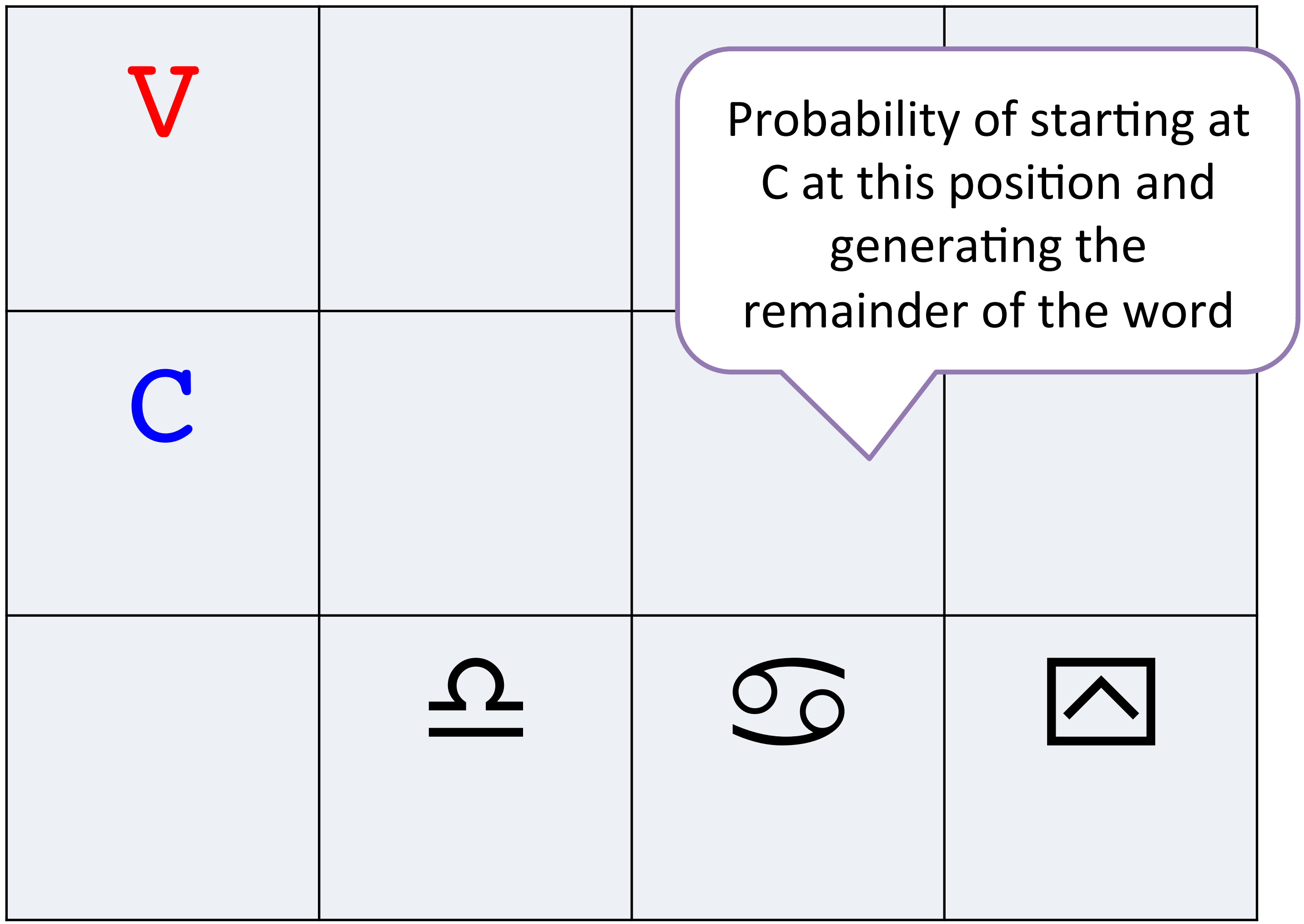
P(♎|V)=0.01

P(♋|C)=0.02

P(♋|V)=0.14

P(⍓|C)=0.07

P(⍓|V)=0.20



Backward Algorithm

Given these probabilities:

P(C|#)=0.7

P(C|C)=0.4

P(V|V)=0.1

P(♎|C)=0.08

P(♎|V)=0.01

P(♋|C)=0.02

P(♋|V)=0.14

P(⍓|C)=0.07

P(⍓|V)=0.20

V			1
C			1
	♎	♋	⍓

Backward Algorithm

Given these probabilities:

P(C|#)=0.7

P(C|C)=0.4

P(V|V)=0.1

P(♎|C)=0.08

P(♎|V)=0.01

P(♋|C)=0.02

P(♋|V)=0.14

P(⍓|C)=0.07

P(⍓|V)=0.20

V			1
C			1
	♎	♋	⍓

P(C|C)*P(⍓|C)*1
= 0.4*0.07*1 = 0.028

P(V|C)*P(⍓|V)*1
= 0.6*0.20*1 = 0.12

Backward Algorithm

Given these probabilities:
P(C|#)=0.7
P(C|C)=0.4
P(V|V)=0.1

P(♎|C)=0.08
P(♎|V)=0.01
P(♋|C)=0.02
P(♋|V)=0.14
P(⍓|C)=0.07
P(⍓|V)=0.20

V			1
		.148	1
	♎	♋	⍓

P(C|V)*P(⍓|C)*1
= 0.9*0.07*1 = 0.063
P(V|V)*P(⍓|V)*1
= 0.1*0.20*1 = 0.02

Backward Algorithm

Given these probabilities:
P(C|#)=0.7
P(C|C)=0.4
P(V|V)=0.1

P(♎|C)=0.08
P(♎|V)=0.01
P(♋|C)=0.02
P(♋|V)=0.14
P(⍓|C)=0.07
P(⍓|V)=0.20

V		0.083	1
C		0.148	1
		♋	⍓

P(C|C)*P(♋|C)*0.148
= 0.4*0.02*0.148 = 0.001184
P(V|C)*P(♋|V)*0.083
= 0.6*0.14*0.083 = 0.006972

Backward

P(C|V)*P(♋|C)*0.148
= 0.9*0.02*0.148 = 0.002664
P(V|V)*P(♋|V)*0.083
= 0.1*0.14*0.083 = 0.001162

Given these probabilities:
P(C|#)=0.7
P(C|C)=0.4
P(V|V)=0.1

P(♎|C)=0.08
P(♎|V)=0.01
P(♋|C)=0.02
P(♋|V)=0.14
P(⍓|C)=0.07
P(⍓|V)=0.20

V		0.083	1
C	0.00816	0.148	1
	♎	♋	⍓

Backward Algorithm

Total probability of ♎♋⍓ :

P(C|#)
*P(♎|C)
*0.00816
+
P(V|#)
*P(♎|V)
*0.00383
= 0.00045696
+ 0.00001149
= 0.000468

V	0.00383	0.083	1
C	0.00816	0.148	1
	♎	♋	⍓

Forward Prob = Backward Prob

V	0.003	0.004746	1.55×10^{-4}
C	0.056	0.000502	3.13×10^{-4}
#	d	a	y

Total Probability of ♎♋⍓ =

4.68×10^{-4}

Soft Expectation Maximization

1. Guess parameters of model
2. Tag observations with best state sequence
3. Re-estimate model parameters by counting
4. Repeat: go to step 2.

Bad guess can corner model into bad local maximum

Instead of getting counts from the best state sequence, get **expected counts** (weighted average) from all possible state sequences with their probabilities

Forward Backward Algorithm

- How to avoid enumerating all exponentially many state sequences in the E-Step?
  - Dynamic Programming magic!

Emission Counts

- What is the total probability of *all* state sequences where state_2 is C given ♎♋⍓?
- Gives expected number of times C → ♋ at state_2

C	C	C
C	C	V
V	C	V
V	C	C

♎♋⍓

P(day, CCC)
+ P(day, CCV)
+ P(day, VCV)
+ P(day, VCC)

Emission Counts

V	0.003	0.004746	1.55×10^{-4}
C	0.056	0.000502	3.13×10^{-4}
	♎	♋	⍓

$$P(♎♋⍓, \text{state}_2 = C)$$
$$= \text{Forward}(\text{state}_2 = C)$$
$$* \text{ Backward}(\text{state}_2 = C)$$

$$= 0.000502$$
$$* \ 0.148$$
$$= 7.4296 \times 10^{-5}$$

V	0.00383	0.083	1
C	0.00816	0.148	1
	♎	♋	⍓

$$P(\text{state}_2 = C \mid ♎♋⍓)$$
$$= \frac{P(♎♋⍓, \text{state}_2 = C)}{P(♎♋⍓)}$$

$$= 7.4296 \times 10^{-5} / 4.68 \times 10^{-4}$$
$$= \mathbf{0.1588}$$

Emission Counts

- What is the total probability of *all* state sequences where state$_2$ is V given ♎♋⍓?
- Gives expected number of times V → ♋ at state$_2$

C	V	C
C	V	V
V	V	V
V	V	C

♎♋⍓

P(day, CVC)
+ P(day, CVV)
+ P(day, VVV)
+ P(day, VVC)

Emission Counts

V	0.003	0.004746	1.55×10^{-4}
C	0.056	0.000502	3.13×10^{-4}
	♎	♋	⍓

V	0.00383	0.083	1
C	0.00816	0.148	1
	♎	♋	⍓

P(♎♋⍓, $state_2$ = V)
= Forward($state_2$ = V)
* Backward($state_2$ = V)

= 0.004746
* 0.083
= 3.939×10^{-4}

P($state_2$ = V | ♎♋⍓)
= P(♎♋⍓, $state_2$ = V) / P(♎♋⍓)

= $3.939 \times 10^{-4} / 4.68 \times 10^{-4}$
= **0.8412**

Emission Counts

V	0.003	0.004746	1.55×10^{-4}
C	0.056	0.000502	3.13×10^{-4}
	♎	♋	⍓

P(♎♋⍓, $state_2$ = V)
= Forward($state_2$ = V)
* Backward($state_2$ = V)

= 0.004746
* 0.083
= 3.939×10^{-4}

V	0.00383	0.083	1

Sanity Check #2:
P($state_i$=C|obs) +P($state_i$=V|obs) = 1

P($state_2$ = V|♎♋⍓)

$$= \frac{P(♎♋⍓, state_2 = V)}{P(♎♋⍓)}$$

= $3.939 \times 10^{-4} / 4.68 \times 10^{-4}$
= **0.8412**

Expected Emission Counts

- To get expected number of times V → ♋
  - For every string in corpus, at all positions *i* where ♋ occurs, compute total conditional probability of V at position *i*
  - Add up these probabilities

How about transition counts?

- What is the total probability of all state sequences where state_1 is C and state_2 is V given ♎♋⍓?
- Gives expected number of times C → V from state_1 to state_2

C V C
C V V

♎♋⍓

P(day, CVC)
+ P(day, CVV)

Transition Counts

V	0.003	0.004746	$1.55x10^{-4}$
C	0.056	0.000502	$3.13x10^{-4}$
	♎	♋	⍓

V	0.00383	0.083	1
C	0.00816	0.148	1
	♎	♋	⍓

P(♎♋⍓, state$_1$ = C, state$_2$ = V)
= Forward(state$_1$ = C)
* P(V|C) * P(♋|V)
* Backward(state$_2$ = V)

= 0.056
* 0.6 * 0.14
* 0.083 = $3.904x10^{-4}$

P(state$_1$ = C, state$_2$ = V| ♎♋⍓)
= P(♎♋⍓, state$_1$ = C, state$_2$ = V)/P(♎♋⍓)
= $3.904x10^{-4}$/$4.68x10^{-4}$
= 0.8341

Transition Counts

V	0.003	0.004746	1.55×10^{-4}
C	0.056	0.000502	3.13×10^{-4}
	♎	♋	⍓

V	0.00383	0.083	1
C	0.00816	0.148	1
	♎	♋	⍓

P(♎♋⍓,
$state_1$ = C, $state_2$ = C)
= Forward($state_1$ = C)
* P(C|C) * P(♋|C)
* Backward($state_2$ = C)

= 0.056
* 0.4 * 0.02
* 0.148 = 6.6304×10^{-5}

P($state_1$ = C, $state_2$ = C|
♎♋⍓)
= P(♎♋⍓, $state_1$ = C,
$state_2$ = C)/P(♎♋⍓)
= $6.6304 \times 10^{-5} / 4.68 \times 10^{-4}$
= 0.1417

Transition Counts

V	0.003	0.004746	$1.55\text{x}10^{-4}$
C	0.056	0.000502	$3.13\text{x}10^{-4}$
	♎	♋	⍓

V	0.00383	0.083	1
C	0.00816	0.148	1
	♎	♋	⍓

P(♎♋⍓, state_1 = V, state_2 = C)
= Forward(state_1 = V)
* P(C|V) * P(♋|C)
* Backward(state_2 = C)

= 0.003
* 0.9 * 0.02
* 0.148 = $7.992\text{x}10^{-6}$

P(state_1 = V, state_2 = C| ♎♋⍓)
= P(♎♋⍓, state_1 = V, state_2 = C)/P(♎♋⍓)
= $7.992\text{x}10^{-6}/4.68\text{x}10^{-4}$
= 0.0171

Transition Counts

V	0.003	0.004746	1.55×10^{-4}
C	0.056	0.000502	3.13×10^{-4}
	♎	♋	⍓

V	0.00383	0.083	1
C	0.00816	0.148	1
	♎	♋	⍓

P(♎♋⍓, $state_1$ = V, $state_2$ = V)
= Forward($state_1$ = V)
* P(V|V) * P(♋|V)
* Backward($state_2$ = V)

= 0.003
* 0.1 * 0.14
* 0.083 = 3.486×10^{-6}

P($state_1$ = V, $state_2$ = V| ♎♋⍓)
= P(♎♋⍓, $state_1$ = V, $state_2$ = V)/P(♎♋⍓)
= $3.486 \times 10^{-6} / 4.68 \times 10^{-4}$
= 0.0074

Transition Counts Sanity Check

- $P(\text{state}_1 = C, \text{state}_2 = C \mid$ ♎♋⍓$) = 0.1417$
- $P(\text{state}_1 = C, \text{state}_2 = V \mid$ ♎♋⍓$) = 0.8341$
- $P(\text{state}_1 = V, \text{state}_2 = V \mid$ ♎♋⍓$) = 0.0074$
- $P(\text{state}_1 = V, \text{state}_2 = C \mid$ ♎♋⍓$) = 0.0171$

Sanity Check #3:
These should add to 1

Expected Transition Counts

- To get expected number of times C → V
  - For every string in corpus, at all positions *i* compute total conditional probability of C at position *i* and V at position *i+1*
  - Add up these probabilities

What are HMMs used for?

- *Any* sequence modeling problem!
  - Part of speech tagging
  - Speech recognition
  - Machine translation
  - Gene prediction in computational biology
  - Modeling human gait or facial expressions
  - Financial analysis

Techniques are even more general

- Dynamic Programming
  - Shortest path in graphs
    (map route-finding, network routing)
  - Audio alignment
  - AI game-playing
  - Fast matrix multiplication

Techniques are even more general

- Expectation Maximization:
  any problem to find unknown structure of data
  - Clustering
  - Computer vision
  - Signal processing
  - Financial modeling

Variants

<table>
<tr><th></th><th>Discrete States</th><th>Continuous States</th></tr>
<tr><td>Discrete Observations</td><td>HMMs with Discrete Emission Probabilities (Text, Biology)</td><td rowspan="2">Kalman Filters, Particle Filters (Finance, GPS, Astronomy)</td></tr>
<tr><td>Continuous Observations</td><td>HMMs with Continuous Emission Probabilities (Speech, Vision)</td></tr>
</table>

Dynamic Bayesian Networks

- HMMs assume each observation symbol is generated from a single state
- What if there's more to the story?
  - Word generated from part-of-speech, topic, sentiment, previous word
  - Sound generated from phoneme, voice quality, gender

Machine Learning Fundamentals

- Inference: answering questions about data under an existing model
  - The cross entropy of a text under an n-gram language model
  - The best HMM state sequence for an observation
  - Computing the perplexity of a text under a language model
  - Edit distance two strings under a set of edit costs
  - Best parse of a sentence under a grammar

Machine Learning Fundamentals

- Learning: computing the parameters of a model from a collection of data
  - Estimating an n-gram language model from a corpus
  - Estimating the channel model spelling error costs from a corpus of spelling mistakes
  - Estimating the HMM transition and emission probabilities
- Our favorite strategy: maximum likelihood