Hidden Markov Models

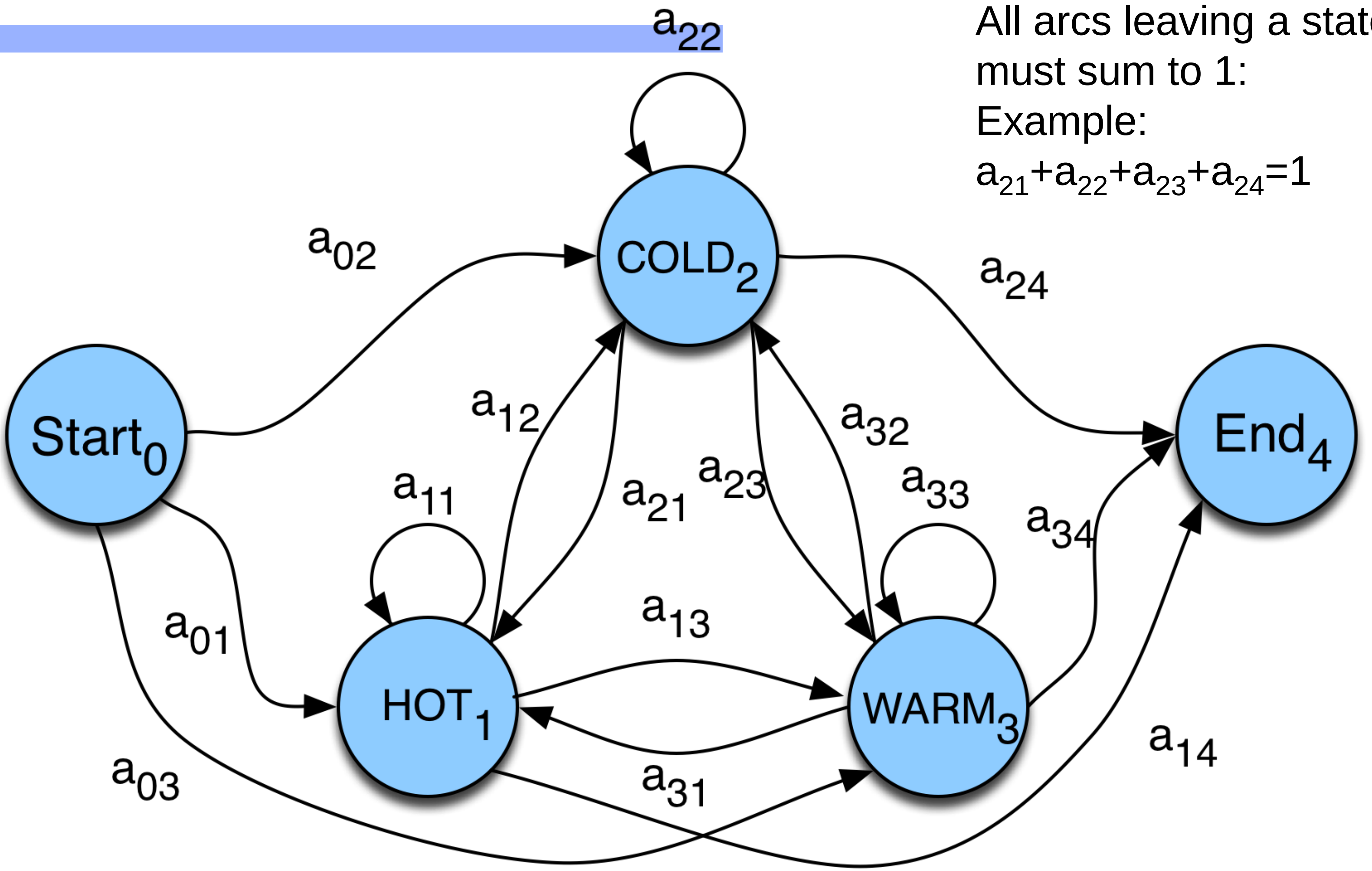
Outline

- Markov Chains
- Hidden Markov Models
- Three Algorithms for HMMs
  - The Forward Algorithm
  - The Viterbi Algorithm
- Applications:
  - The Ice Cream Task
  - Part of Speech Tagging

Definitions

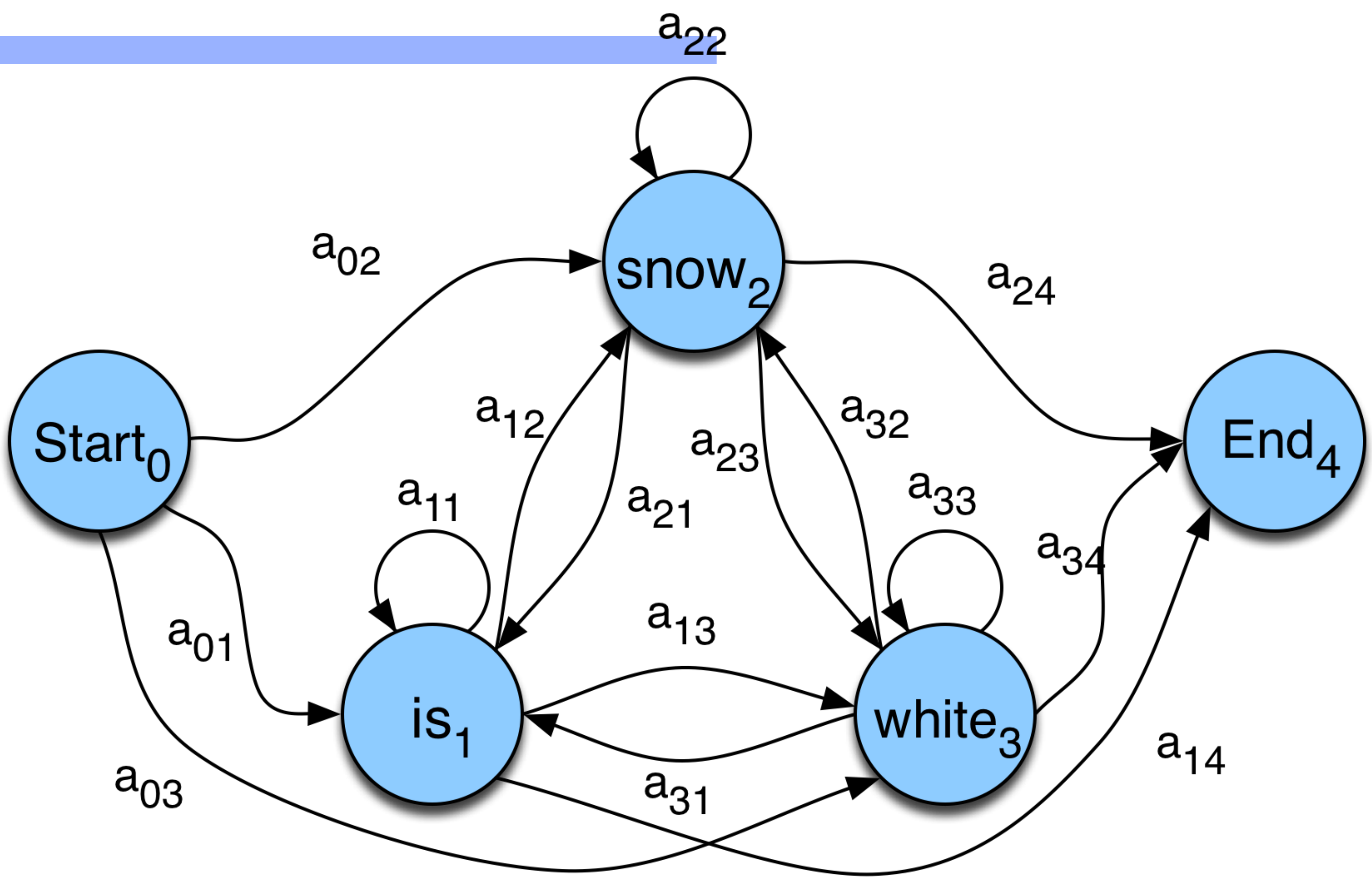
- A **weighted finite-state automaton**
  - **An FSA with** probabilities on the arcs
  - The sum of the probabilities leaving any arc <u>must sum to one</u>
- A **Markov chain (or observable Markov Model)**
  - a special case of a WFST in which the input sequence uniquely determines which states the automaton will go through
- Markov chains can't represent inherently ambiguous problems
  - Useful for assigning probabilities to unambiguous sequences

Markov chain for weather



All arcs leaving a state must sum to 1:
Example:
$a_{21}+a_{22}+a_{23}+a_{24}=1$

Markov chain for words



Markov chain = “First-order observable Markov Model”

- a set of states
  - $Q = q_1, q_2 \ldots q_{N;}$ the state at time t is q_t
- Transition probabilities:
  - a set of probabilities $A = a_{01} a_{02} \ldots a_{n1} \ldots a_{nn}$.
  - Each a_{ij} represents the probability of transitioning from state i to state j
  - The set of these is the transition probability matrix A

$$a_{ij} = P(q_t = j \mid q_{t-1} = i) \quad 1 \le i, j \le N$$

$$\sum_{j=1}^{N} a_{ij} = 1; \quad 1 \le i \le N$$

- Distinguished start and end states

Markov chain = "First-order observable Markov Model"

- Markov Assumption:

  Current state only depends on previous state

$$P(q_i \mid q_1 \ldots q_{i-1}) = P(q_i \mid q_{i-1})$$

Another representation for start state

- Instead of start state
- Special initial probability vector π
  - An initial distribution over probability of start states

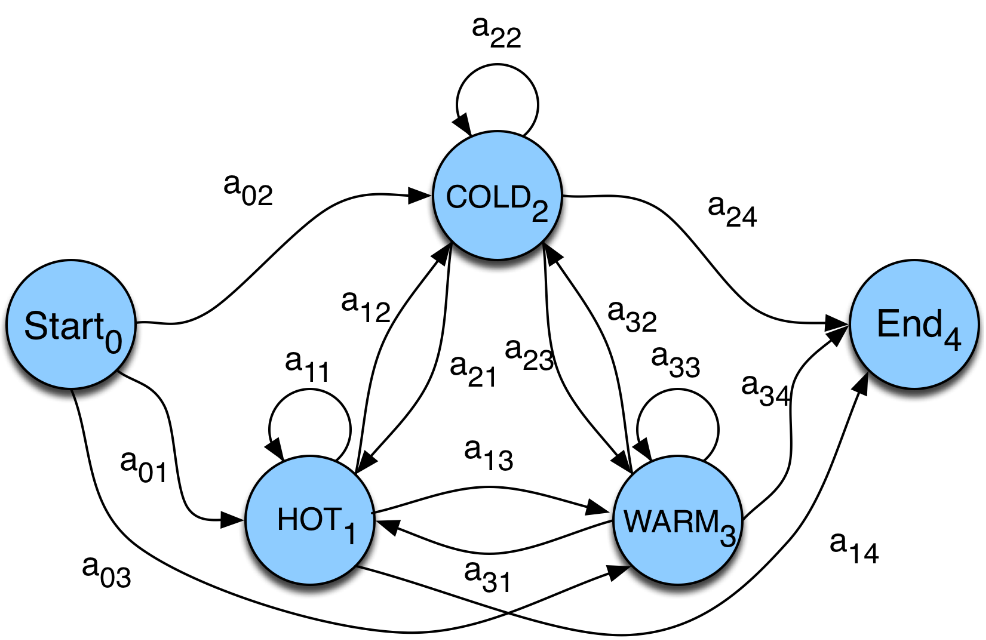
$$\pi_i = P(q_1 = i) \quad 1 \le i \le N$$

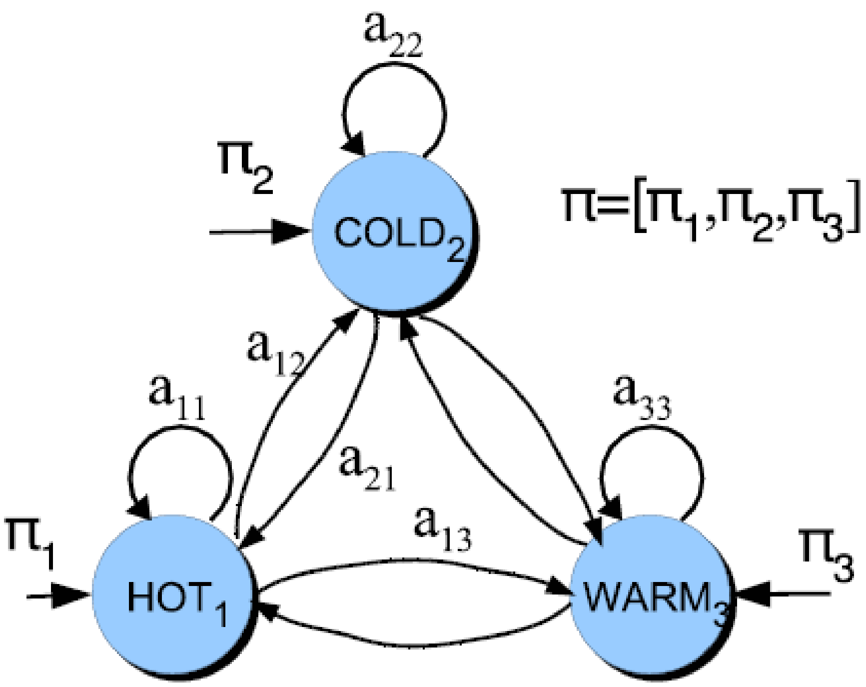
- Constraints:

$$\sum_{j=1}^{N} \pi_j = 1$$

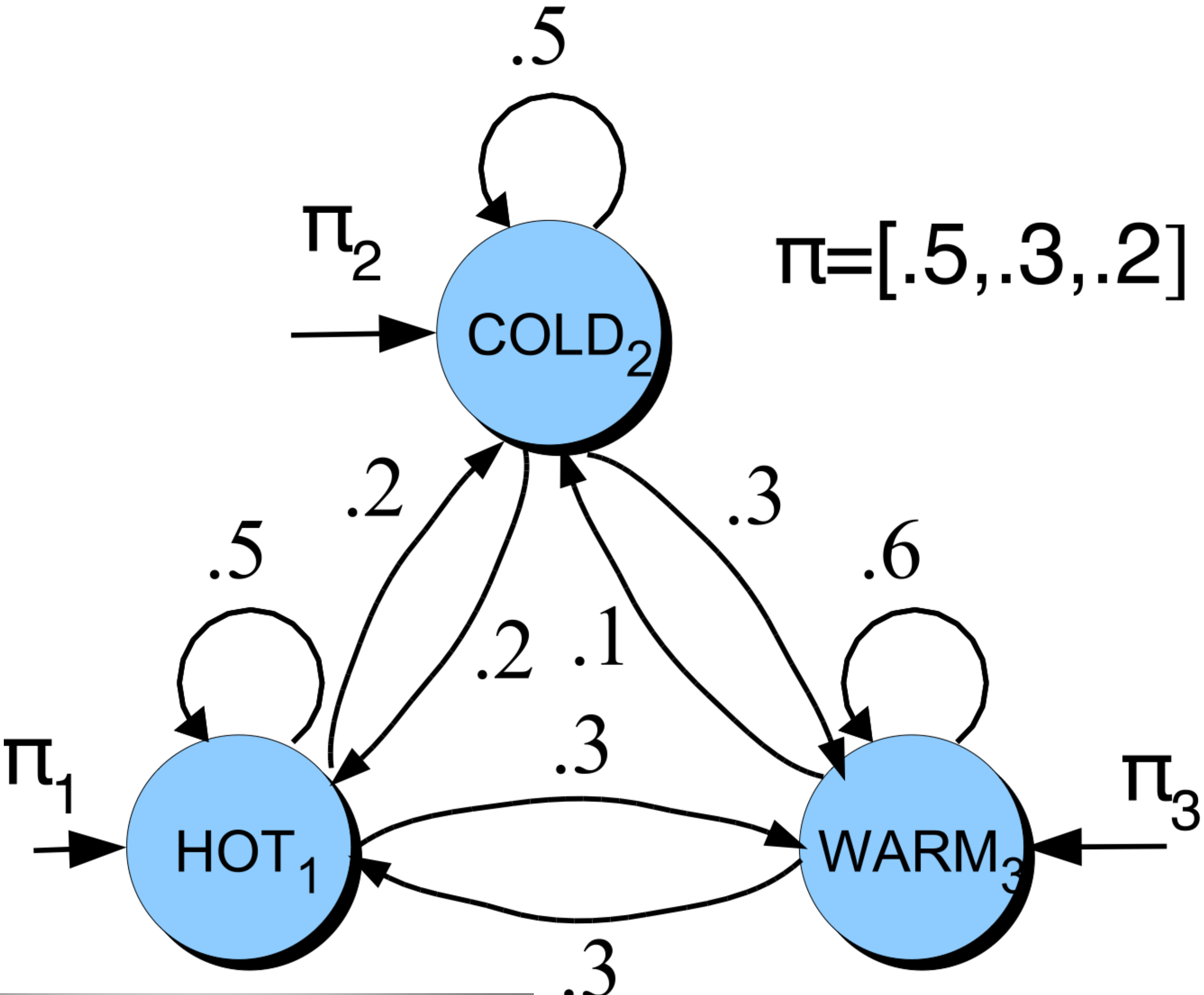
- $QA=q_x,q_y,\ldots$. A set of legal accepting states

The weather model using π

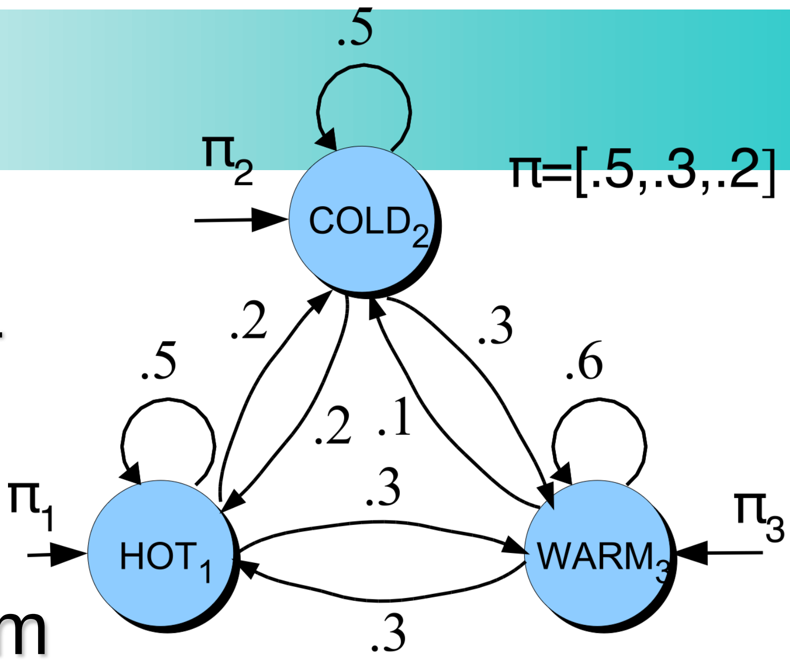




The weather model: specific example



Markov chain for weather



- What is the probability of 4 consecutive warm days?
- Sequence is

  warm-warm-warm-warm
- i.e., state sequence is 3-3-3-3

$$P(3, 3, 3, 3) =$$

$$\pi_3 a_{33} a_{33} a_{33} = 0.2 \text{ x } (0.6)^3 = 0.0432$$

- So far the states are visible
- To model more complex transitions we might need to use hidden markov models where states are hidden and we only make observations.

HMM for Ice Cream

- You are a climatologist in the year 2799
- Studying global warming
- You can't find any records of the weather in Delhi for summer of 2025
- But you find Rahul's diary
- Which lists how many ice-creams Rahul ate every date that summer
- Our job: figure out how hot it was

Hidden Markov Model

- For Markov chains, the output symbols are the same as the states.
  - See **hot** weather: we're in state **hot**
- But in named-entity or part-of-speech tagging (and speech recognition and other things)
  - The output symbols are **words**
  - But the hidden states are something else
    - **Part-of-speech tags**
    - **Named entity tags**
- So we need an extension!
- A Hidden Markov Model is an extension of a Markov chain in which the input symbols are not the same as the states.
- This means we don't know which state we are in.

Hidden Markov Models

$Q = q_1 q_2 \ldots q_N$	a set of N **states**
$A = a_{11} a_{12} \ldots a_{n1} \ldots a_{nn}$	a **transition probability matrix** A, each a_{ij} representing the probability of moving from state i to state j, s.t. $\sum_{j=1}^{n} a_{ij} = 1 \quad \forall i$
$O = o_1 o_2 \ldots o_T$	a sequence of T **observations**, each one drawn from a vocabulary $V = v_1, v_2, ..., v_V$
$B = b_i(o_t)$	a sequence of **observation likelihoods**, also called **emission probabilities**, each expressing the probability of an observation o_t being generated from a state i
q_0, q_F	a special **start state** and **end (final) state** that are not associated with observations, together with transition probabilities $a_{01} a_{02} \ldots a_{0n}$ out of the start state and $a_{1F} a_{2F} \ldots a_{nF}$ into the end state

Assumptions

- **Markov assumption:**

$$P(q_i \mid q_1 \ldots q_{i-1}) = P(q_i \mid q_{i-1})$$

  - the current state is dependent only on the previous state.
  - this represents the memory of the model

- **Output-independence assumption**

$$P(o_t \mid O_1^{t-1}, q_1^t) = P(o_t \mid q_t)$$

  - the output observation at time t is dependent only on the current state
  - it is independent of previous observations and states

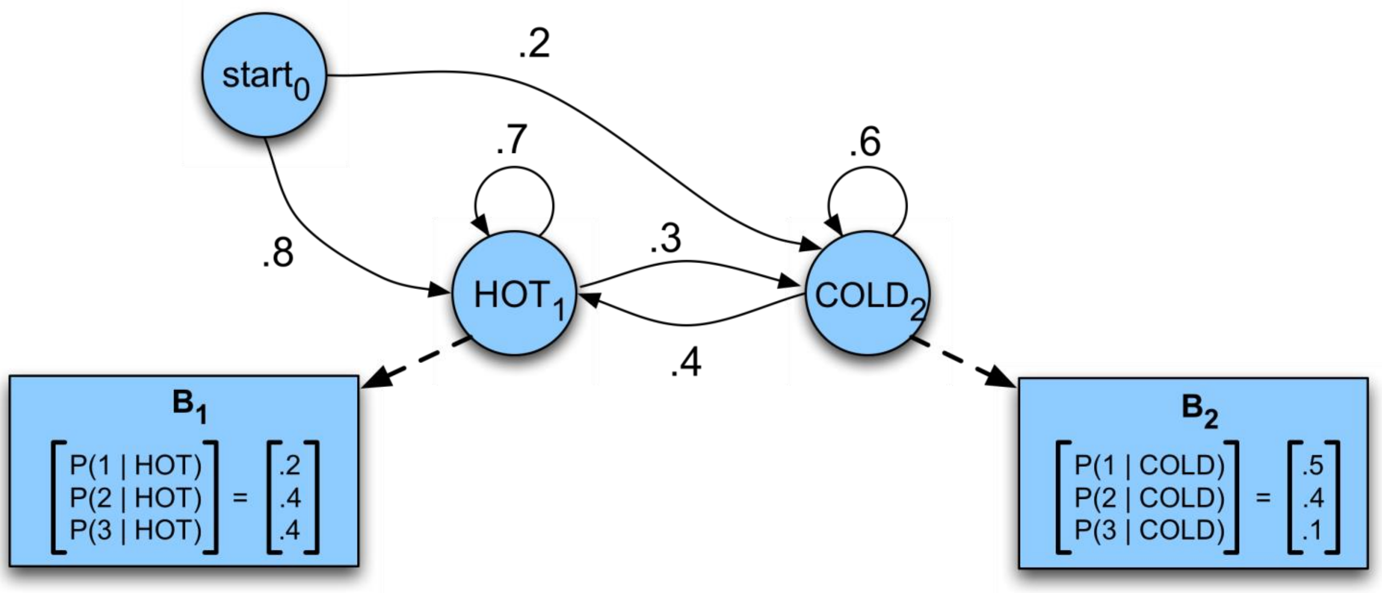
The Ice Cream task (cont.)

- Given a sequence of observations O,
  - each observation an integer = number of ice creams eaten
  - Figure out correct hidden sequence Q of weather states (H or C) which caused Rahul to eat the ice cream

In other words:
Given:
    Ice Cream Observation Sequence: 1,2,3,2,2,2,3…
Produce:
    Weather Sequence: H,C,H,H,H,C…

An HMM for this task

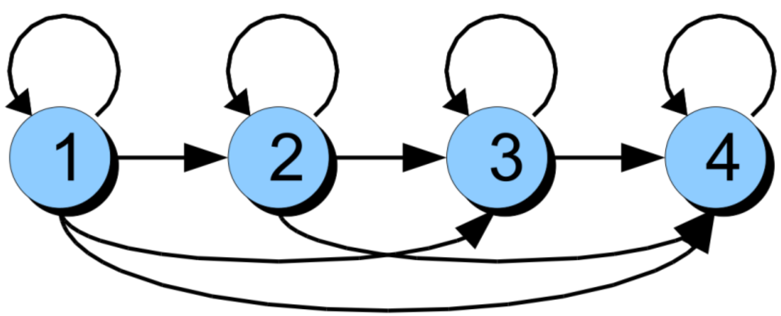
- Relating numbers of ice creams eaten by Rahul (the observations) to the weather (the hidden variables)



Different types of HMM structure

Left-right or Bakis mode:

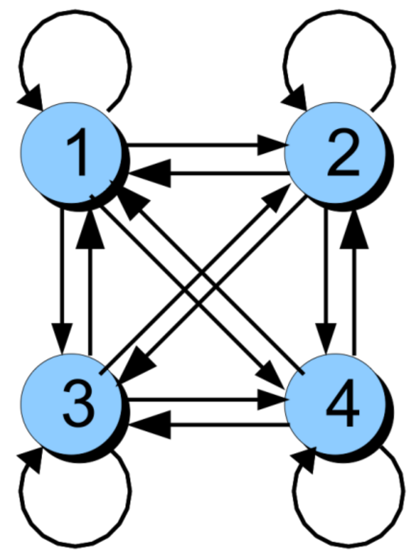
- no transitions are allowed to states whose indices are lower than the current state $a_{ij} = 0; \forall j < i$
- Left-right models are best suited to model signals whose properties change over time, such as speech
- When using left-right models, some additional constraints are commonly placed, such as preventing large transitions $a_{ij} = 0 \ \forall j > i + \Delta$



Bakis = left-to-right

Ergodic

- a fully connected model
- each state can be reached in one step from every other state
- most general type of HMM



Ergodic = fully-connected

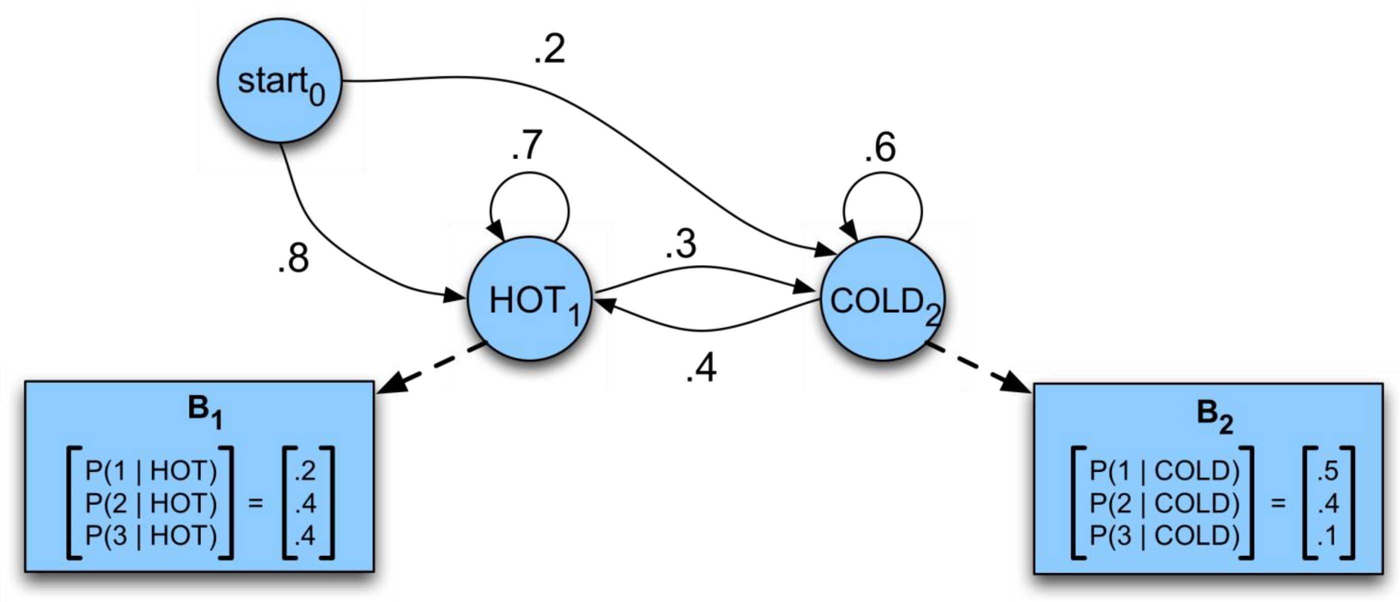
The Three Basic Problems for HMMs : more formally

- Problem 1 (**Evaluation**): Given the observation sequence $O=(o_1 o_2 \ldots o_T)$, and an HMM model $\Phi = (A,B)$, how do we efficiently compute $P(O|\,\Phi)$, the probability of the observation sequence, given the model
- Problem 2 (**Decoding**): Given the observation sequence $O=(o_1 o_2 \ldots o_T)$, and an HMM model $\Phi = (A,B)$, how do we choose a corresponding state sequence $Q=(q_1 q_2 \ldots q_T)$ that is optimal in some sense (i.e., best explains the observations)
- Problem 3 (**Learning**): How do we adjust the model parameters $\Phi = (A,B)$ to maximize $P(O|\,\Phi\,)$?

Problem 1: computing the observation likelihood

Computing Likelihood: Given an HMM $\lambda = (A,B)$ and an observation sequence O, determine the likelihood $P(O|\lambda)$.

- Given the following HMM:



- How likely is the sequence 3 1 3?

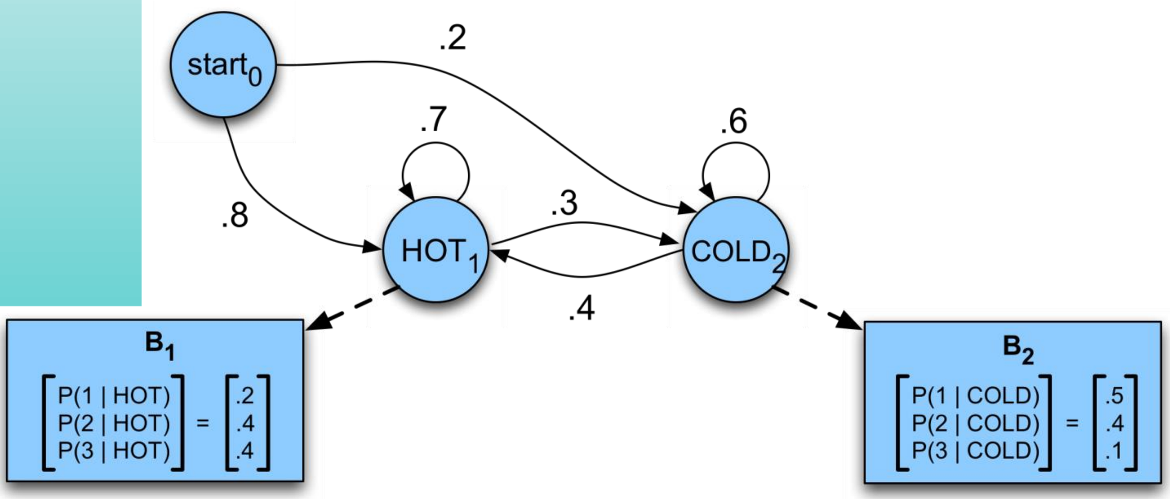
How to compute likelihood

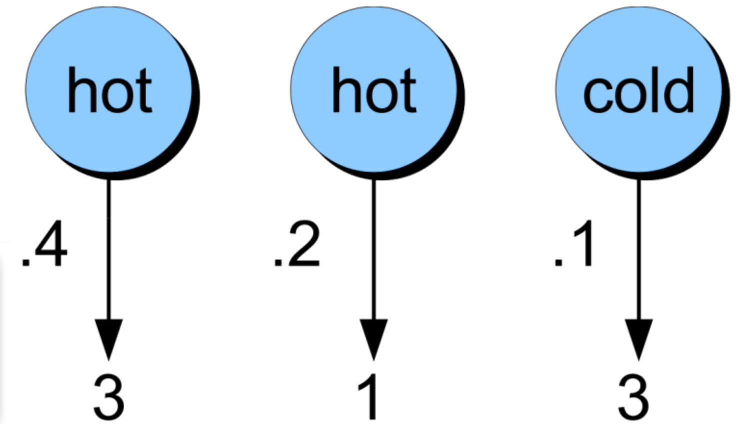
- For a Markov chain, we just follow the states 3 1 3 and multiply the probabilities
- But for an HMM, we don't know what the states are!
- So let's start with a simpler situation.
- Computing the observation likelihood for a **given** hidden state sequence
  - Suppose we knew the weather and wanted to predict how much ice cream Rahul would eat.
  - i.e. $P(3\ 1\ 3 \mid \mathrm{H\ H\ C})$

Computing likelihood of 3 1 3 given hidden state sequence

$$P(O|Q) = \prod_{i=1}^{T} P(o_i|q_i)$$

$$P(3\ 1\ 3|\text{hot hot cold}) \;=\; P(3|\text{hot}) \times P(1|\text{hot}) \times P(3|\text{cold})$$

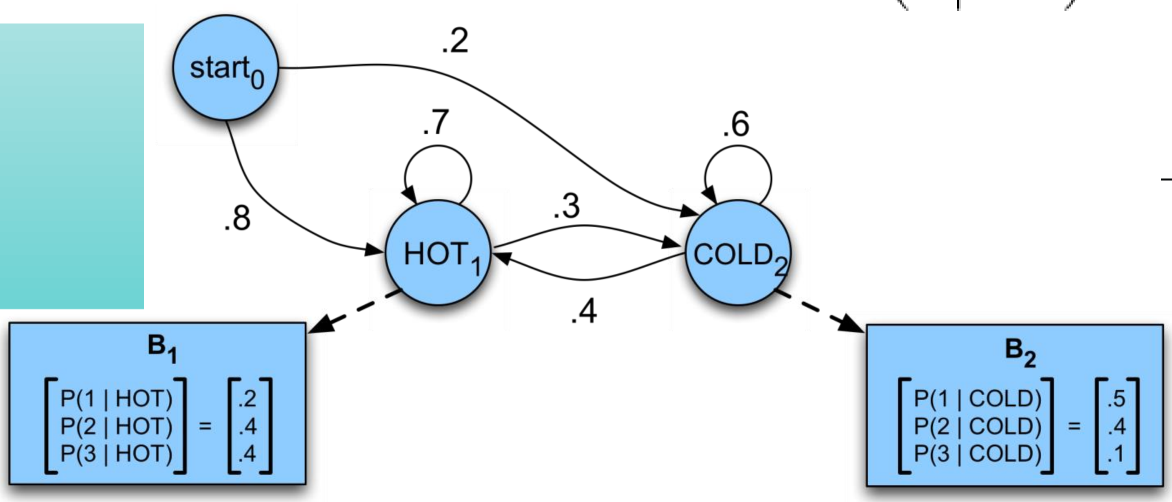


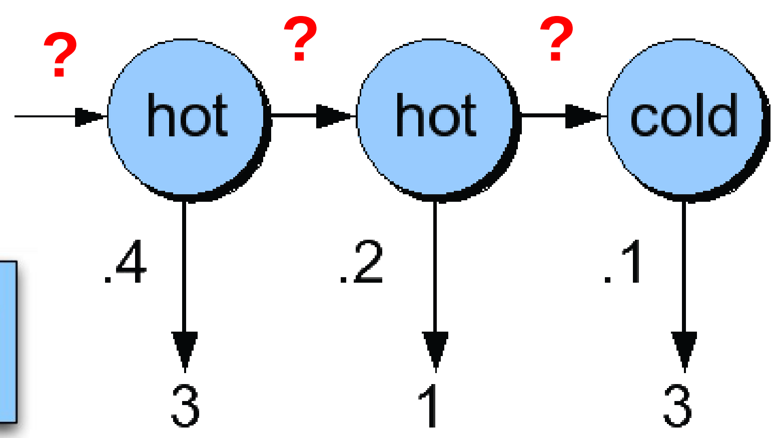


Computing joint probability of observation and state sequence

$$P(O,Q) = P(O|Q) \times P(Q) = \prod_{i=1}^{n} P(o_i|q_i) \times \prod_{i=1}^{n} P(q_i|q_{i-1})$$

$$\begin{aligned} P(3\ 1\ 3, \text{hot hot cold}) &= P(\text{hot}|\text{start}) \times P(\text{hot}|\text{hot}) \times P(\text{cold}|\text{hot}) \\ &\times P(3|\text{hot}) \times P(1|\text{hot}) \times P(3|\text{cold}) \end{aligned}$$





Computing total likelihood of 3 1 3

- We would need to sum over
  - Hot hot cold
  - Hot hot hot
  - Hot cold hot
  - ….

$$P(O) = \sum_Q P(O,Q) = \sum_Q P(O|Q)P(Q)$$

- How many possible hidden state sequences are there for this sequence?

$$P(3\ 1\ 3) = P(3\ 1\ 3, \text{cold cold cold}) + P(3\ 1\ 3, \text{cold cold hot}) + P(3\ 1\ 3, \text{hot hot cold}) + \dots$$

- How about in general for an HMM with N hidden states and a sequence of T observations?
  - N^T
- So we can't just do separate computation for each hidden state sequence.

Instead: the Forward algorithm

- A kind of **dynamic programming** algorithm
  - Just like Minimum Edit Distance
  - Uses a table to store intermediate values
- Idea:
  - Compute the likelihood of the observation sequence
  - By summing over all possible hidden state sequences
  - But doing this efficiently
    - By folding all the sequences into a single **trellis**

The forward algorithm

- The goal of the forward algorithm is to compute

$$P(o_1, o_2 \ldots o_T, q_T = q_F \mid \lambda)$$

- We'll do this by recursion

The forward algorithm

- Each cell of the forward algorithm trellis $\alpha_t(j)$
  - Represents the probability of being in state j
  - After seeing the first t observations
  - Given the automaton
- Each cell thus expresses the following probability

$$\alpha_t(j) = P(o_1, o_2 \ldots o_t, q_t = j \mid \lambda)$$

The Forward Recursion

1. Initialization:

$$\alpha_1(j) \ = \ a_{0j}b_j(o_1) \ \ 1 \leq j \leq N$$

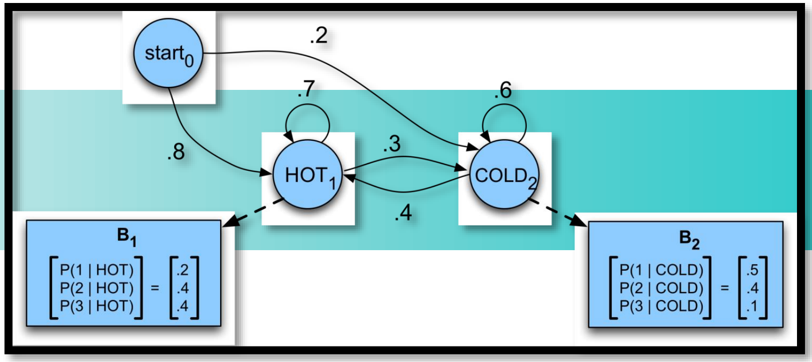
2. Recursion (since states 0 and F are non-emitting):

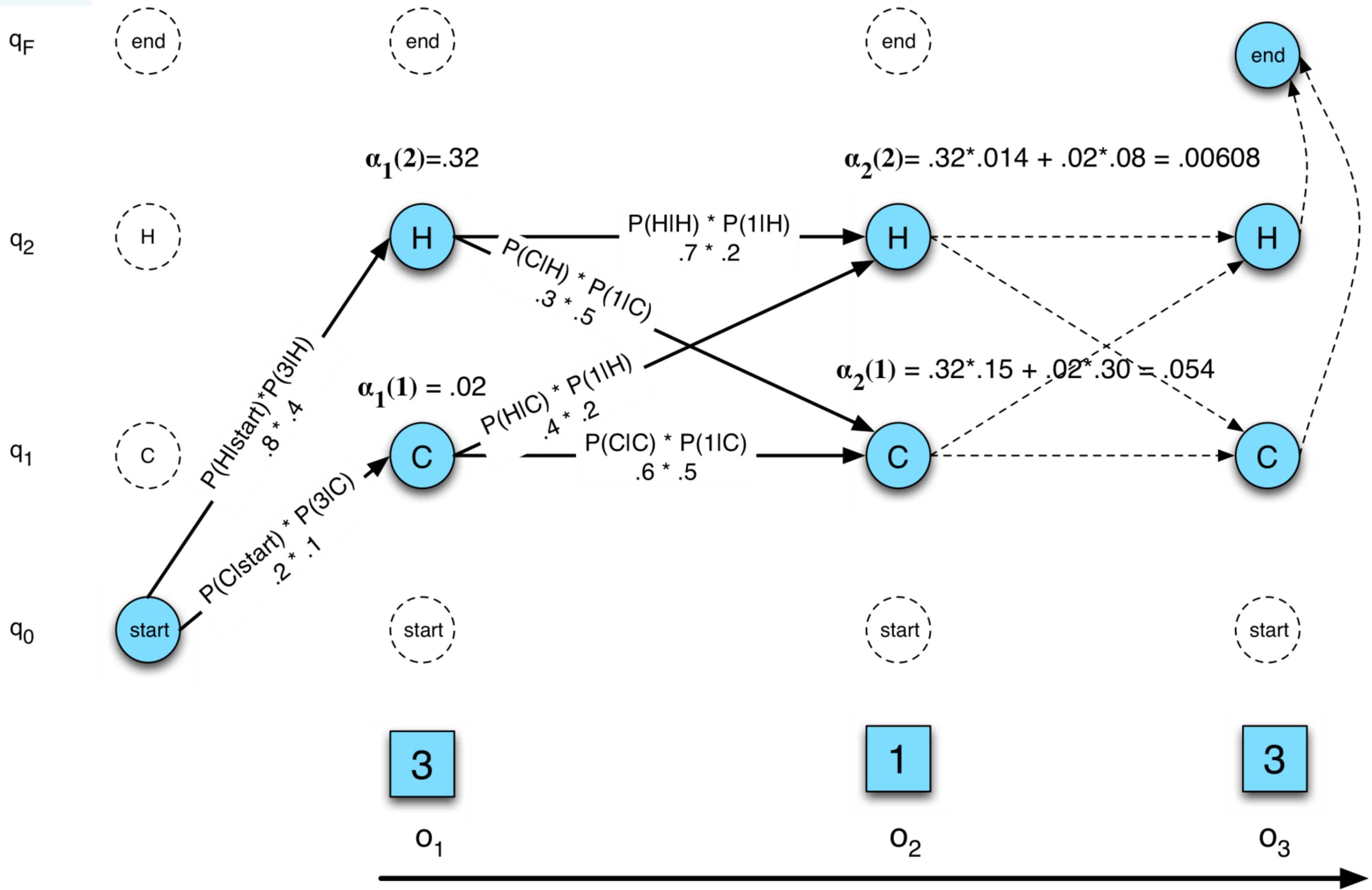
$$\alpha_t(j) = \sum_{i=1}^{N} \alpha_{t-1}(i)a_{ij}b_j(o_t); \quad 1 \leq j \leq N, 1 < t \leq T$$

3. Termination:

$$P(O|\lambda) = \alpha_T(q_F) = \sum_{i=1}^{N} \alpha_T(i)\, a_{iF}$$

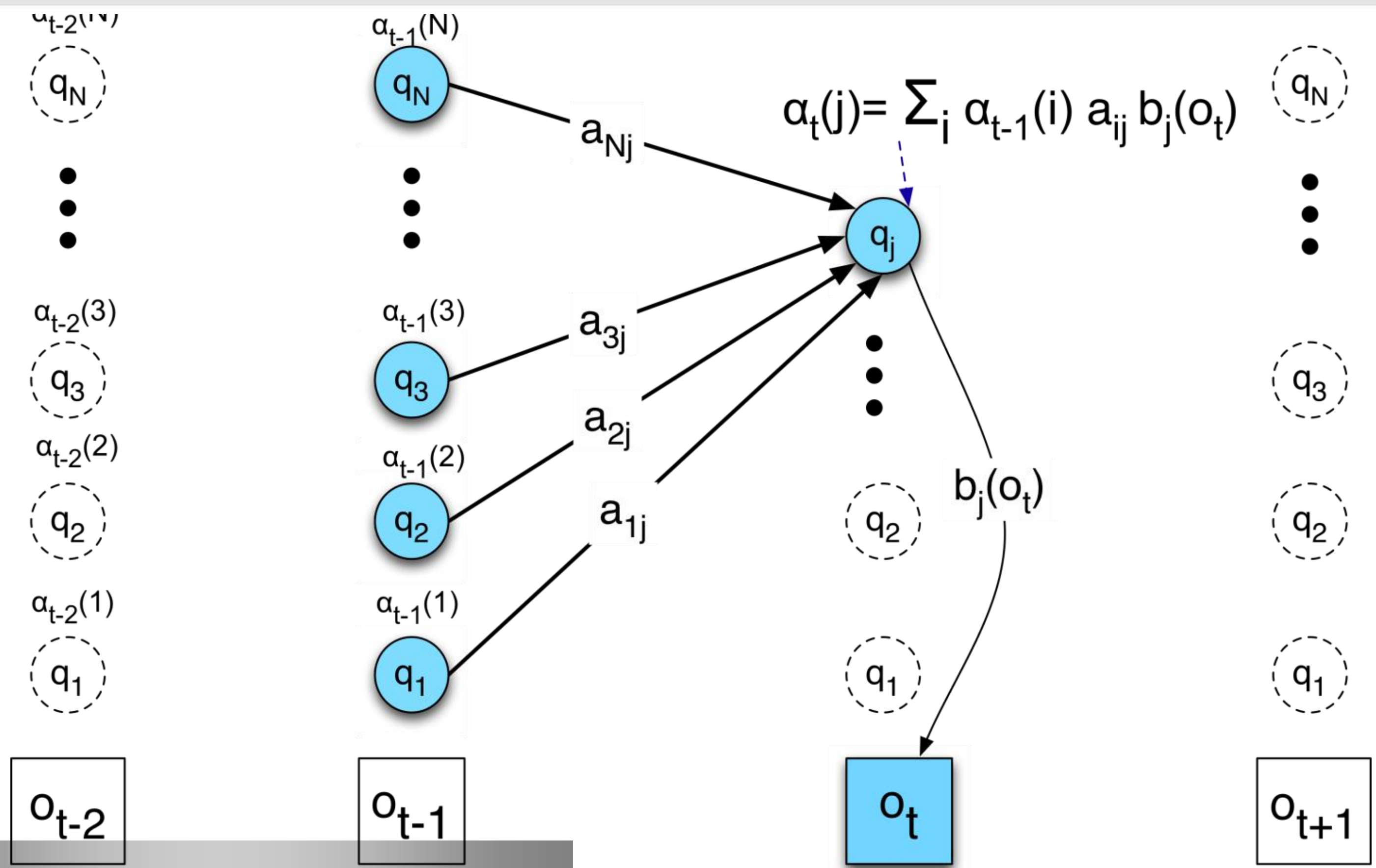
The Forward Trellis





We update each cell

$\alpha_{t-1}(i)$	the **previous forward path probability** from the previous time step
a_{ij}	the **transition probability** from previous state q_i to current state q_j
$b_j(o_t)$	the **state observation likelihood** of the observation symbol o_t given the current state j



The Forward Algorithm

function FORWARD(*observations* of len T, *state-graph* of len N) **returns** *forward-prob*

create a probability matrix *forward[N+2,T]*

for each state s **from** 1 **to** N **do** ; initialization step

$forward[s,1] \leftarrow a_{0,s} * b_s(o_1)$

for each time step t **from** 2 **to** T **do** ; recursion step

for each state s **from** 1 **to** N **do**

$$forward[s,t] \leftarrow \sum_{s'=1}^{N} forward[s',t-1] * a_{s',s} * b_s(o_t)$$

$$forward[q_F,\mathrm{T}] \leftarrow \sum_{s=1}^{N} forward[s,T] * a_{s,q_F}$$; termination step

return $forward[q_F,T]$

Decoding

- Given an observation sequence
  - 3 1 3
- And an HMM
- The task of the **decoder**
  - To find the best **hidden** state sequence
- Given the observation sequence $O=(o_1 o_2 \ldots o_T)$, and an HMM model $\Phi = (A,B)$, how do we choose a corresponding state sequence $Q=(q_1 q_2 \ldots q_T)$ that is optimal in some sense (i.e., best explains the observations)

Decoding

- One possibility:
  - For each hidden state sequence Q
    - HHH, HHC, HCH,
  - Compute $P(O|Q)$
  - Pick the highest one
- Why not?
  - N^T
- Instead:
  - The Viterbi algorithm
  - Is again a **dynamic programming** algorithm
  - Uses a similar trellis to the Forward algorithm

Viterbi intuition

- We want to compute the joint probability of the observation sequence together with the best state sequence

$$\max_{q_0,q_1,\ldots,q_T} P(q_0,q_1,\ldots,q_T,o_1,o_2,\ldots,o_T,q_T=q_F \mid \lambda)$$

$$v_t(j) = \max_{q_0,q_1,\ldots,q_{t-1}} P(q_0,q_1\ldots q_{t-1},o_1,o_2\ldots o_t,q_t=j|\lambda)$$

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

Viterbi Recursion

1. **Initialization:**

$$
\begin{aligned}
v_1(j) &= a_{0j} b_j(o_1) \;\; 1 \le j \le N \\
bt_1(j) &= 0
\end{aligned}
$$

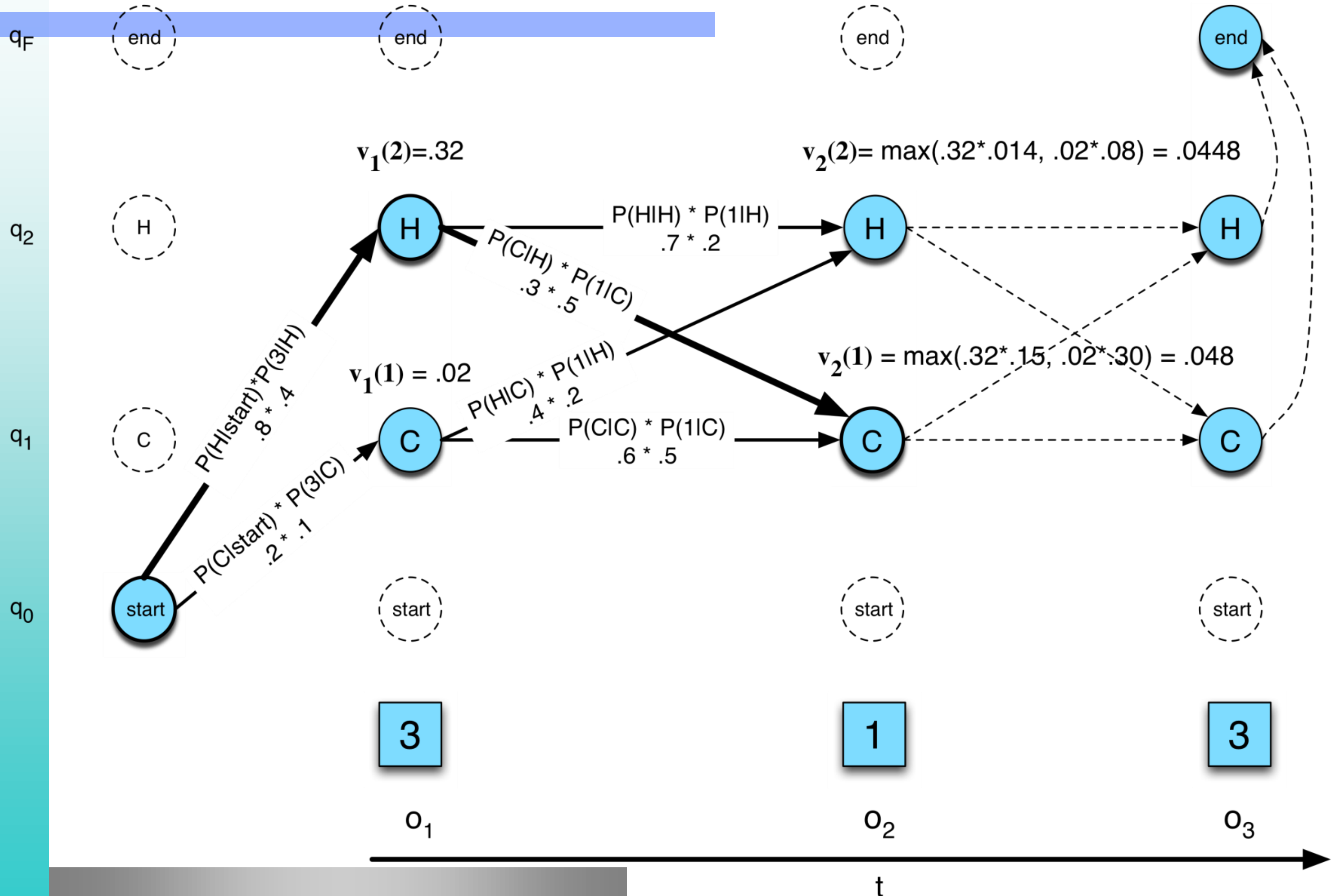
2. **Recursion** (recall that states 0 and q_F are non-emitting):

$$
\begin{aligned}
v_t(j) &= \max_{i=1}^{N} v_{t-1}(i)\, a_{ij}\, b_j(o_t); \quad 1 \le j \le N, 1 < t \le T \\
bt_t(j) &= \operatorname*{argmax}_{i=1}^{N} v_{t-1}(i)\, a_{ij}\, b_j(o_t); \quad 1 \le j \le N, 1 < t \le T
\end{aligned}
$$

3. **Termination:**

$$
\begin{aligned}
\text{The best score:} \quad P* = v_t(q_F) &= \max_{i=1}^{N} v_T(i) * a_{i,F} \\
\text{The start of backtrace:} \quad q_T* = bt_T(q_F) &= \operatorname*{argmax}_{i=1}^{N} v_T(i) * a_{i,F}
\end{aligned}
$$

The Viterbi trellis



Viterbi intuition

- Process observation sequence left to right
- Filling out the trellis
- Each cell:

$$v_t(j) = \max_{q_0,q_1,\ldots,q_{t-1}} P(q_0,q_1 \ldots q_{t-1},o_1,o_2 \ldots o_t,q_t = j|\lambda)$$

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

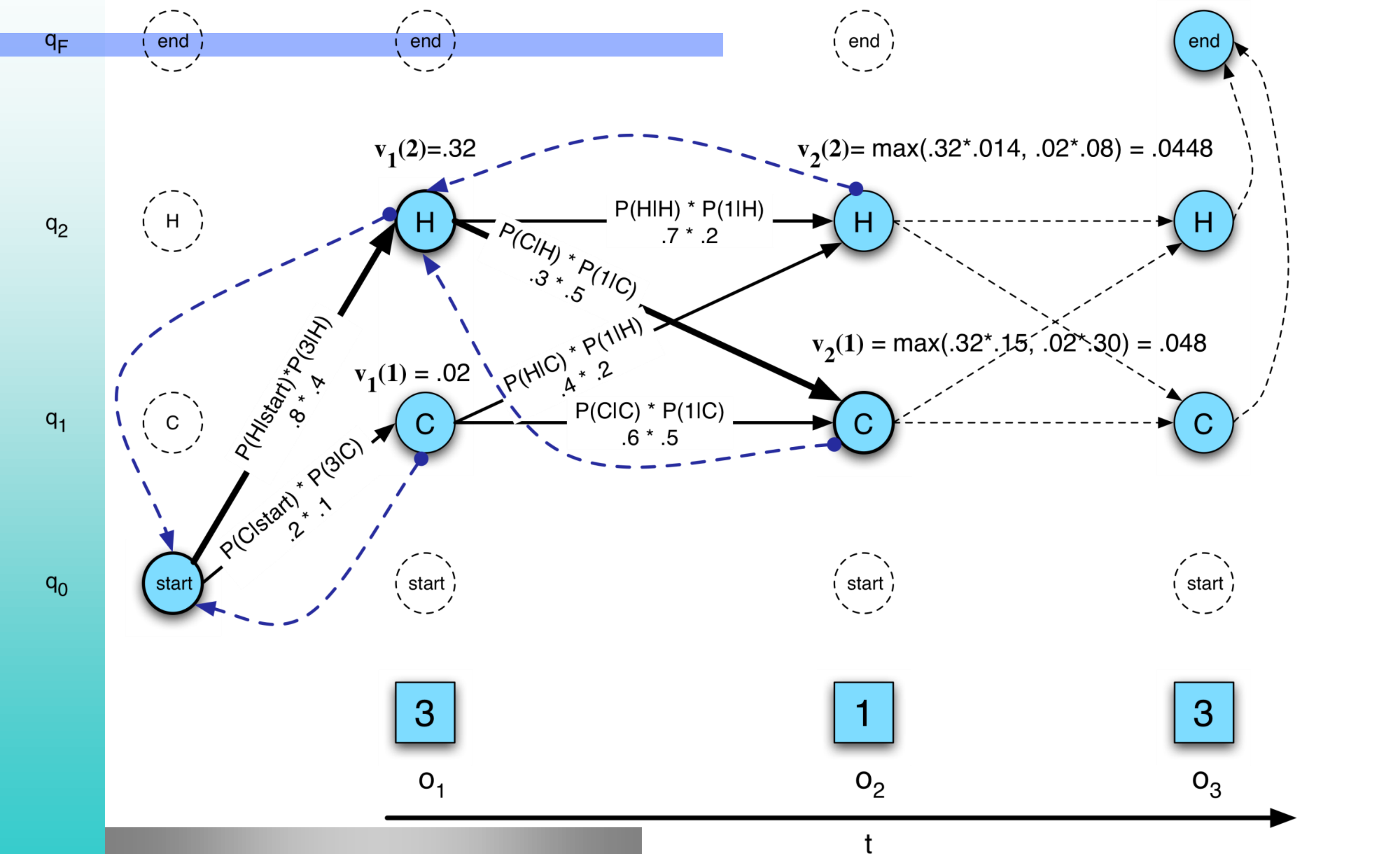
$v_{t-1}(i)$	the **previous Viterbi path probability** from the previous time step
a_{ij}	the **transition probability** from previous state q_i to current state q_j
$b_j(o_t)$	the **state observation likelihood** of the observation symbol o_t given the current state j

Viterbi Algorithm

function VITERBI(*observations* of len T, *state-graph* of len N) **returns** *best-path*

create a path probability matrix *viterbi[N+2,T]*
for each state s **from** 1 **to** N **do** ; initialization step
$viterbi[s,1] \leftarrow a_{0,s} * b_s(o_1)$
$backpointer[s,1] \leftarrow 0$
for each time step t **from** 2 **to** T **do** ; recursion step
for each state s **from** 1 **to** N **do**
$$viterbi[s,t] \leftarrow \max_{s'=1}^{N} viterbi[s',t-1] * a_{s',s} * b_s(o_t)$$
$$backpointer[s,t] \leftarrow \operatorname*{argmax}_{s'=1}^{N} viterbi[s',t-1] * a_{s',s}$$
$viterbi[q_F,T] \leftarrow \max_{s=1}^{N} viterbi[s,T] * a_{s,q_F}$; termination step
$backpointer[q_F,T] \leftarrow \operatorname*{argmax}_{s=1}^{N} viterbi[s,T] * a_{s,q_F}$; termination step
return the backtrace path by following backpointers to states back in time from $backpointer[q_F,T]$

Viterbi backtrace



Learning

On iPad

Hidden Markov Models for Part of Speech Tagging

Part of speech tagging

- 8 (ish) traditional parts of speech
  - Noun, verb, adjective, preposition, adverb, article, interjection, pronoun, conjunction, etc
  - This idea has been around for over 2000 years (Dionysius Thrax of Alexandria, c. 100 B.C.)
  - Called: parts-of-speech, lexical category, word classes, morphological classes, lexical tags, POS
  - We'll use POS most frequently
  - I'll assume that you all know what these are

POS examples

N	noun	*chair, bandwidth, pacing*
V	verb	*study, debate, munch*
ADJ	adj	*purple, tall, ridiculous*
ADV	adverb	*unfortunately, slowly,*
P	preposition	*of, by, to*
PRO	pronoun	*I, me, mine*
DET	determiner	*the, a, that, those*

POS Tagging example

WORD	tag
the	**DET**
koala	**N**
put	**V**
the	**DET**
keys	**N**
on	**P**
the	**DET**
table	**N**

POS Tagging

- Words often have more than one POS: *back*
  - The *back* door = JJ (adjective)
  - On my *back* = NN
  - Win the voters *back* = RB (adverb)
  - Promised to *back* the bill = VB
- The POS tagging problem is to determine the POS tag for a particular instance of a word.

POS tagging as a sequence classification task

- We are given a sentence (an “observation” or “sequence of observations”)
  - Secretariat is expected to race tomorrow
  - She promised to back the bill
- What is the best sequence of tags which corresponds to this sequence of observations?
- Probabilistic view:
  - Consider all possible sequences of tags
  - Out of this universe of sequences, choose the tag sequence which is most probable given the observation sequence of n words $w_1 \ldots w_n$.

Getting to HMM

- We want, out of all sequences of n tags $t_1 \ldots t_n$ the single tag sequence such that $P(t_1 \ldots t_n | w_1 \ldots w_n)$ is highest.

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}}\, P(t_1^n | w_1^n)$$

- Hat ^ means "our estimate of the best one"
- $\text{Argmax}_x\, f(x)$ means "the x such that $f(x)$ is maximized"

Getting to HMM

- This equation is guaranteed to give us the best tag sequence

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} P(t_1^n | w_1^n)$$

- But how to make it operational? How to compute this value?
- Intuition of Bayesian classification:
  - Use Bayes rule to transform into a set of other probabilities that are easier to compute

Using Bayes Rule

$$P(x|y) = \frac{P(y|x)P(x)}{P(y)}$$

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} \frac{P(w_1^n|t_1^n)P(t_1^n)}{P(w_1^n)}$$

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} P(w_1^n|t_1^n)P(t_1^n)$$

Likelihood and prior

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} \overbrace{P(w_1^n|t_1^n)}^{\text{likelihood}} \overbrace{P(t_1^n)}^{\text{prior}}$$

$$P(w_1^n|t_1^n) \approx \prod_{i=1}^{n} P(w_i|t_i)$$

$$P(t_1^n) \approx \prod_{i=1}^{n} P(t_i|t_{i-1})$$

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} P(t_1^n|w_1^n) \approx \underset{t_1^n}{\operatorname{argmax}} \prod_{i=1}^{n} P(w_i|t_i)P(t_i|t_{i-1})$$

Two kinds of probabilities (1)

- Tag transition probabilities $P(t_i|t_{i\text{-}1})$
  - Determiners likely to precede adjs and nouns
    - That/DT flight/NN
    - The/DT yellow/JJ hat/NN
    - So we expect P(NN|DT) and P(JJ|DT) to be high
    - But P(DT|JJ) to be:
  - Compute P(NN|DT) by counting in a labeled corpus:

$$P(t_i|t_{i-1}) = \frac{C(t_{i-1}, t_i)}{C(t_{i-1})}$$

$$P(NN|DT) = \frac{C(DT, NN)}{C(DT)} = \frac{56,509}{116,454} = .49$$

Two kinds of probabilities (2)

- Word likelihood probabilities $p(w_i|t_i)$
  - VBZ (3sg Pres verb) likely to be "is"
  - Compute P(is|VBZ) by counting in a labeled corpus:

$$P(w_i|t_i) = \frac{C(t_i, w_i)}{C(t_i)}$$

$$P(is|VBZ) = \frac{C(VBZ, is)}{C(VBZ)} = \frac{10,073}{21,627} = .47$$

POS tagging: likelihood and prior

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} \overbrace{P(w_1^n|t_1^n)}^{\text{likelihood}} \overbrace{P(t_1^n)}^{\text{prior}}$$

$$P(w_1^n|t_1^n) \approx \prod_{i=1}^{n} P(w_i|t_i)$$

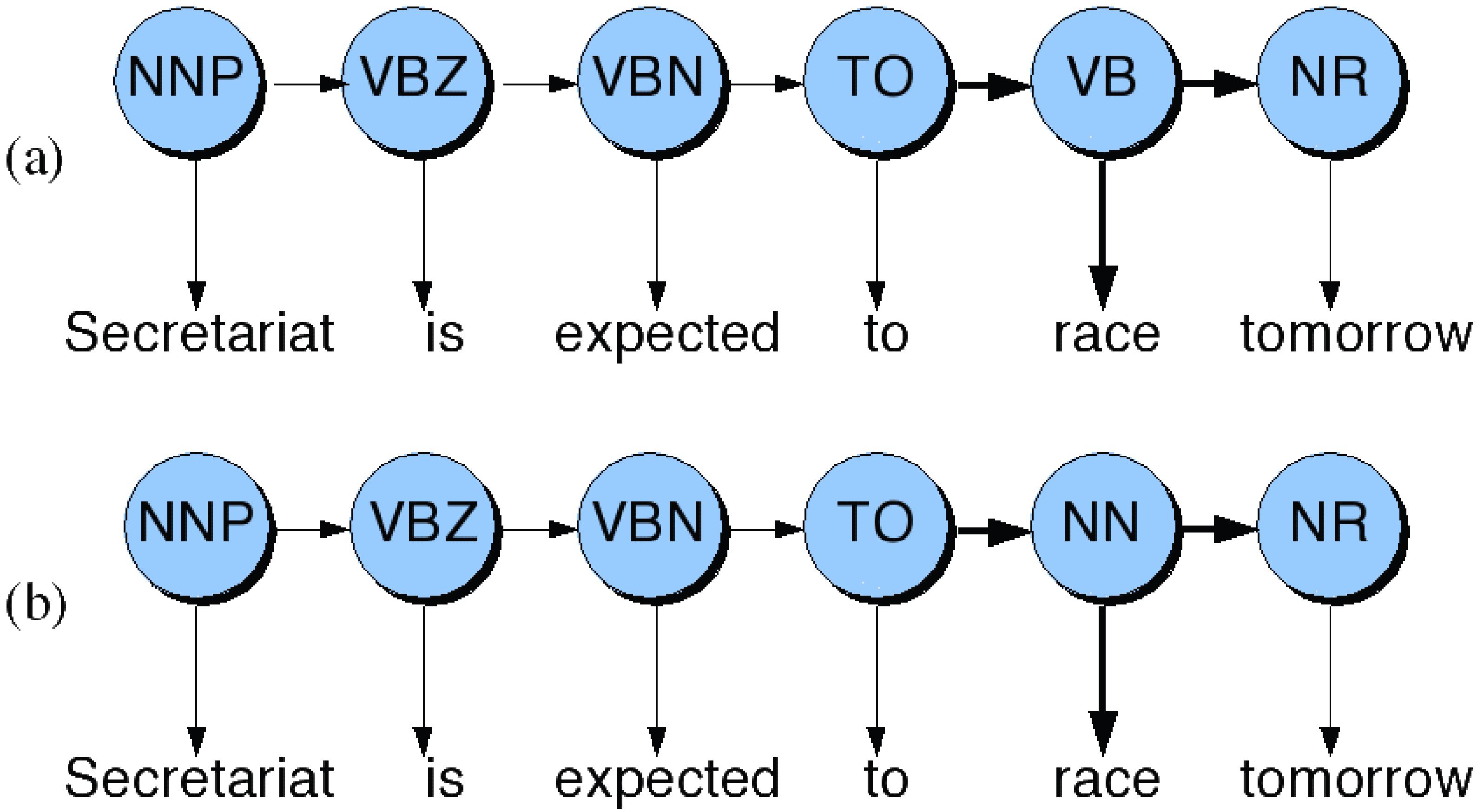
$$P(t_1^n) \approx \prod_{i=1}^{n} P(t_i|t_{i-1})$$

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} P(t_1^n|w_1^n) \approx \underset{t_1^n}{\operatorname{argmax}} \prod_{i=1}^{n} P(w_i|t_i) P(t_i|t_{i-1})$$

An Example: the verb "race"

- Secretariat/NNP is/VBZ expected/VBN to/TO **race**/VB tomorrow/NR
- People/NNS continue/VB to/TO inquire/VB the/DT reason/NN for/IN the/DT **race**/NN for/IN outer/JJ space/NN
- How do we pick the right tag?

Disambiguating "race"

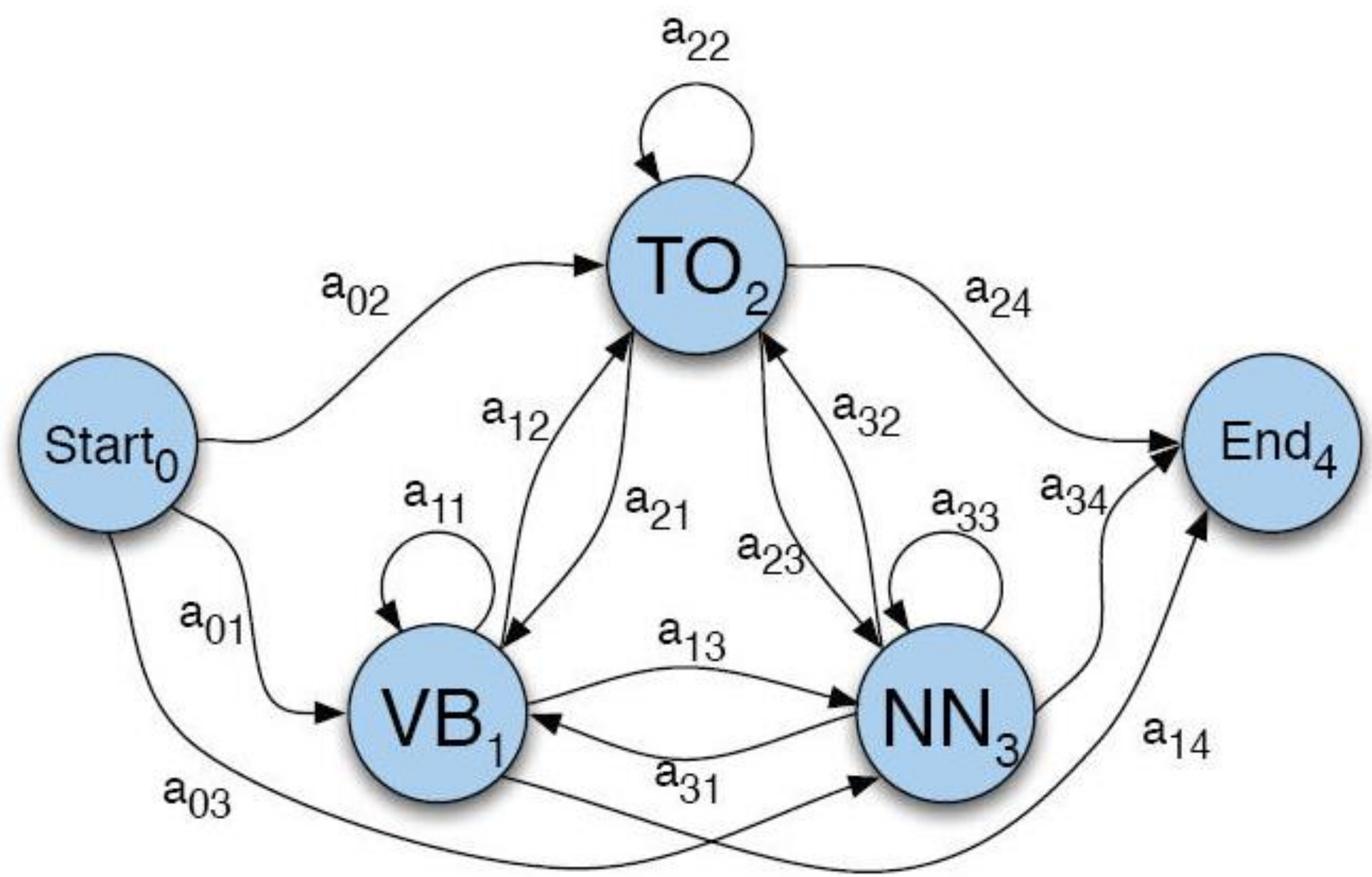


- $P(NN|TO) = .00047$
- $P(VB|TO) = .83$
- $P(race|NN) = .00057$
- $P(race|VB) = .00012$
- $P(NR|VB) = .0027$
- $P(NR|NN) = .0012$

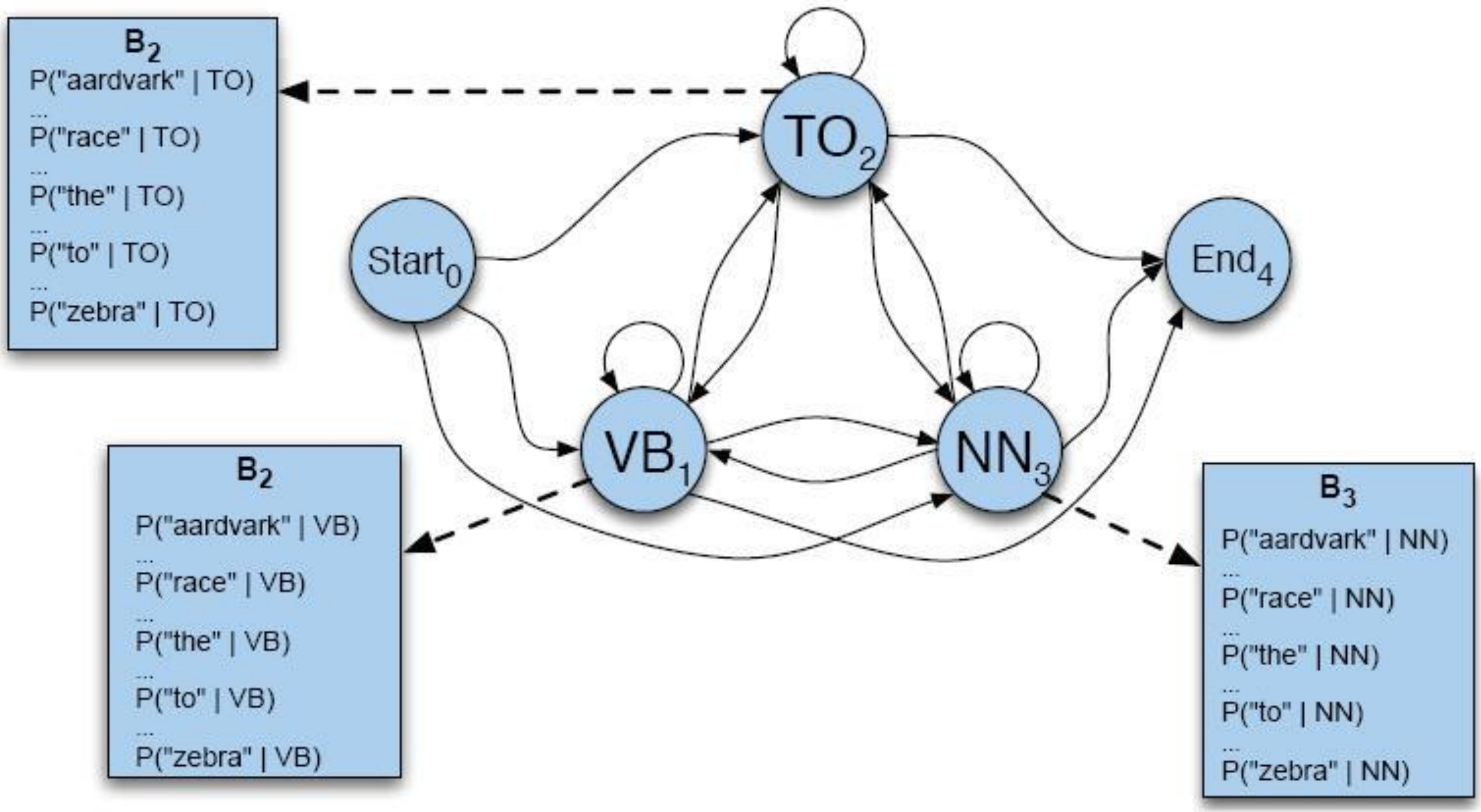
- $P(VB|TO)P(NR|VB)P(race|VB) = .00000027$
- $P(NN|TO)P(NR|NN)P(race|NN)=.00000000032$

- So we (correctly) choose the verb reading

Transitions between the hidden states of HMM, showing A probs



B observation likelihoods for POS HMM



The A matrix for the POS HMM

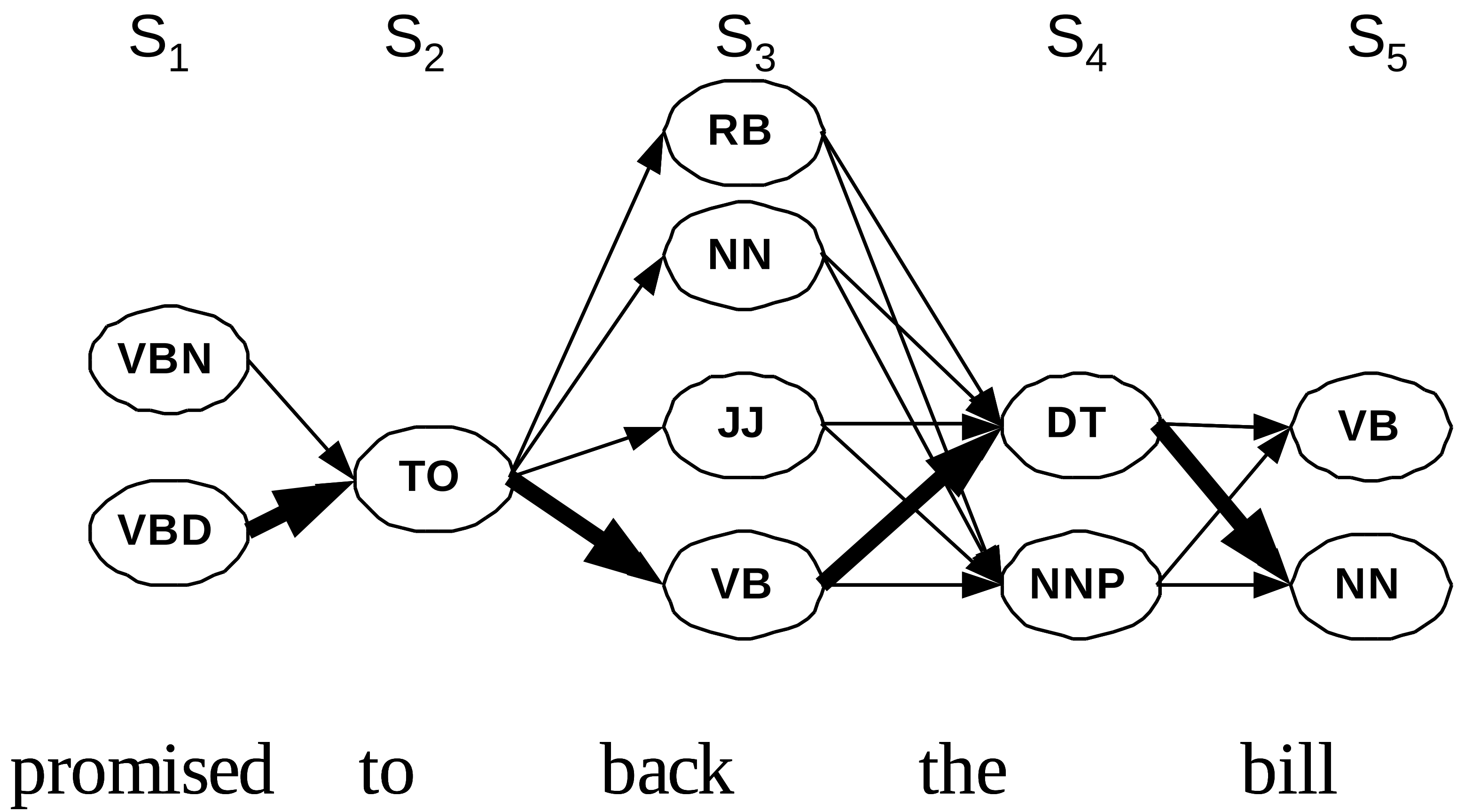
-

	VB	TO	NN	PPSS
<s>	.019	.0043	.041	.067
VB	.0038	.035	.047	.0070
TO	.83	0	.00047	0
NN	.0040	.016	.087	.0045
PPSS	.23	.00079	.0012	.00014

The B matrix for the POS HMM

	I	want	to	race
VB	0	.0093	0	.00012
TO	0	0	.99	0
NN	0	.000054	0	.00057
PPSS	.37	0	0	0

Viterbi intuition: we are looking for the best 'path'



Slide from Dekang Lin

Viterbi example

-