Hidden Markov Models

J&M Appendix A; AIMA 15

CSE 597: Natural Language Processing

Outline

- Formalizing Hidden Markov Models (HMMs)
- Computing Likelihood of a sequence: Forward Algorithm
- Most likely sequence of hidden States: Viterbi Algorithm
- Part-of-Speech (POS) Tagging

Hidden Markov Models

J&M Appendix A; AIMA 15

CSE 597: Natural Language Processing

Formalizing HMMs

Markov Chain versus HMM

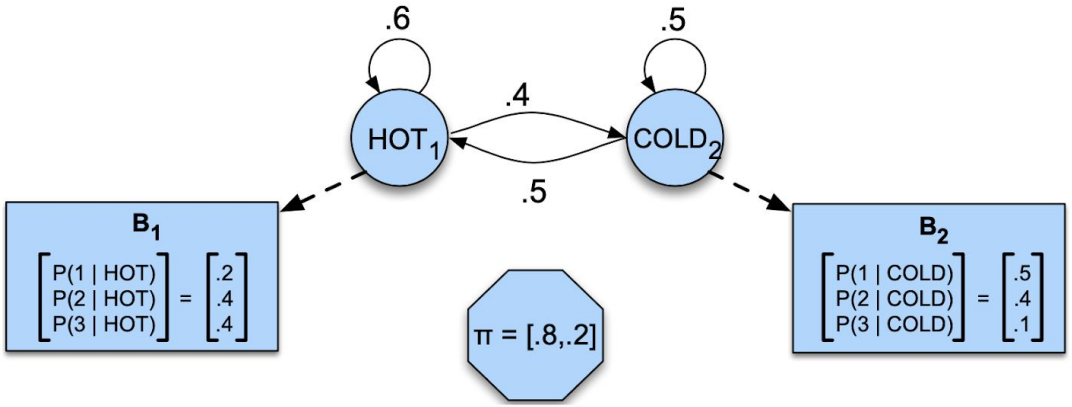
- An HMM is a **non-deterministic** Markov Chain: cannot uniquely identify a state sequence
- States are partially observed (sensor model)

Formal Specification of an HMM

$Q = q_1 q_2 \ldots q_N$	a set of N **states**
$A = a_{11} \ldots a_{ij} \ldots a_{NN}$	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
$\pi = \pi_1, \pi_2, ..., \pi_N$	an **initial probability distribution** over states. π_i is the probability that the Markov chain will start in state i. Some states j may have $\pi_j = 0$, meaning that they cannot be initial states. Also, $\sum_{i=1}^{n} \pi_i = 1$

Figure from Martin & Jurafsky, 3rd Ed., Appendix A

Ice Cream Example



- The only evidence for the weather (hot vs. cold) on a given day is how many ice creams Jason ate that day (Jason Eisner; example from Dan Jurafsky)
  - Q hidden sequence of weather states (H or C)
  - O sequence of observations of number of ice creams

One Possible Sequence of Hidden States (of Eight)

- Given an $\boldsymbol{HMM\ \lambda = (A,B)}$ and an observation sequence $\boldsymbol{O}$, determine $\boldsymbol{P(O|\lambda)}$
- EG: probability of an ice cream sequence 3, 1, 3

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

- $\boldsymbol{P(3,\ 1,\ 3,\ hot,\ hot,\ cold) = P(3|hot)\ x\ P(1|hot)\ x\ P(3|cold)\ x\ P(hot|start)\ x\ P(cold|hot)\ x\ P(hot|cold)}$

Likelihood: Summing the Possible State Sequences

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

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

Likelihood of $O = 3\ 1\ 3$ sums over eight 3-event sequences

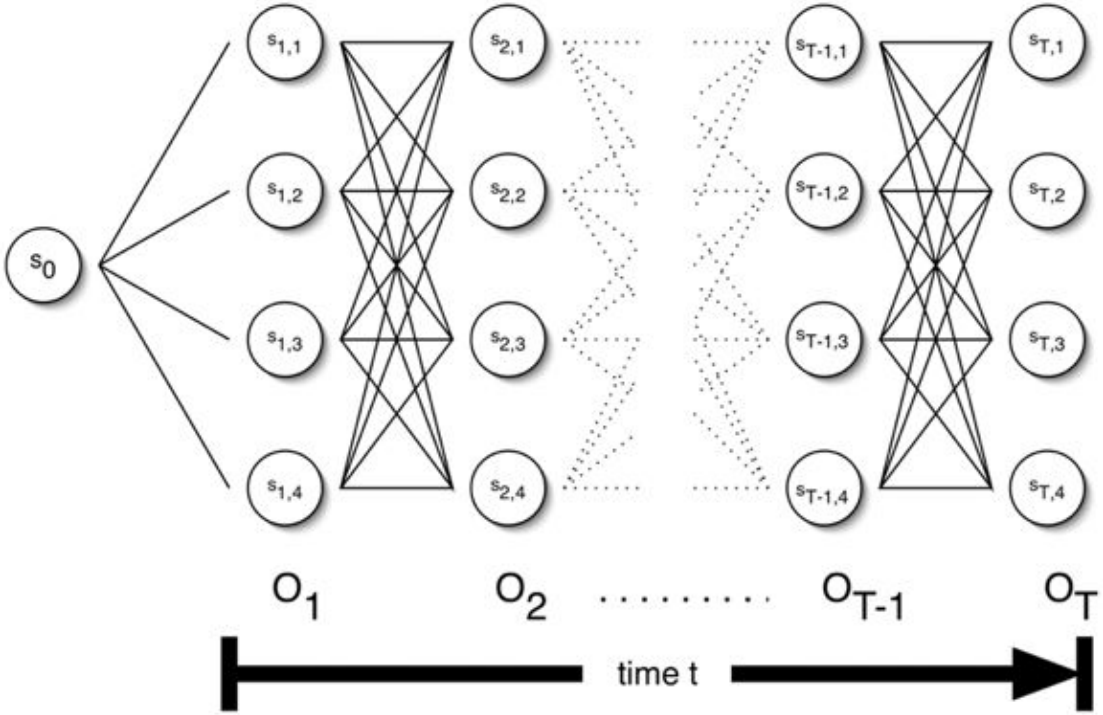
- $P(3\ 1\ 3,\ hot\ cold\ hot) = P(3|hot) \times P(1|cold) \times P(3|hot) \times P(hot|start) \times P(cold|hot) \times P(hot|cold)$
- $P(3\ 1\ 3,\ hot\ hot\ cold) = P(3|hot) \times P(1|hot) \times P(3|cold) \times P(hot|start) \times P(hot|hot) \times P(cold|hot)$

. . .

Motivation for Dynamic Programming

- Calculation of $P(O \mid \lambda)$
  - Sum the probabilities of all possible state sequences in the HMM
  - The probability of each state sequence is the product of the state transitions and emission probabilities
- Naïve computation is very expensive. Given T observations and N states, there are N^T possible state sequences.
  - For $T=10$ and $N=10$, 10 billion different paths!
- Solution: linear time dynamic programming
  - DP: uses a table (trellis) to store intermediate computations

Trellis Data Structure



Size of the Trellis

- N nodes per column, where N is the number of states
- S columns, where S is the length of the sequence
- E edges, one for each transition
- Total trellis size is approximately S(N+E)
  - For $N=10,\ S=10$:
    - $E = (N \times S)$ {edges from S_n to S_{n+1}} $= 10^2$
    - $S(N+E) = 10(10+100) = 1,100 << 10^{10}$

Hidden Markov Models

J&M Appendix A; AIMA 15

CSE 597: Natural Language Processing

Forward Algorithm

Structure of the Forward Algorithm

- **Initialization**: all the ways to start
- **Induction**: all the ways to go from any given state at time t to any subsequent state at time $t+1$
  - Given the probability for state q_i at time t, induction carries forward the probability to each next q_j at time $t+1$
- **Termination**: all the ways to end

Forward Probabilities

- For a given HMM λ, given that the state q is i at time t, what is the probability that the partial observation $o_1 \ldots o_t$ has been generated?

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

- Forward algorithm computes $a_t(i)$ $1 < i < N$, $1 < t < T$ in time $0(N^2T)$ using the trellis, where T is number of observations and N is the number of hidden states

Computing Values of Each Trellis Entry

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 HMM λ . The value of each cell is computed by summing over the probabilities of every path that leads to this cell.

$$\alpha_t(j) = \sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij} b_j(o_t)$$

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

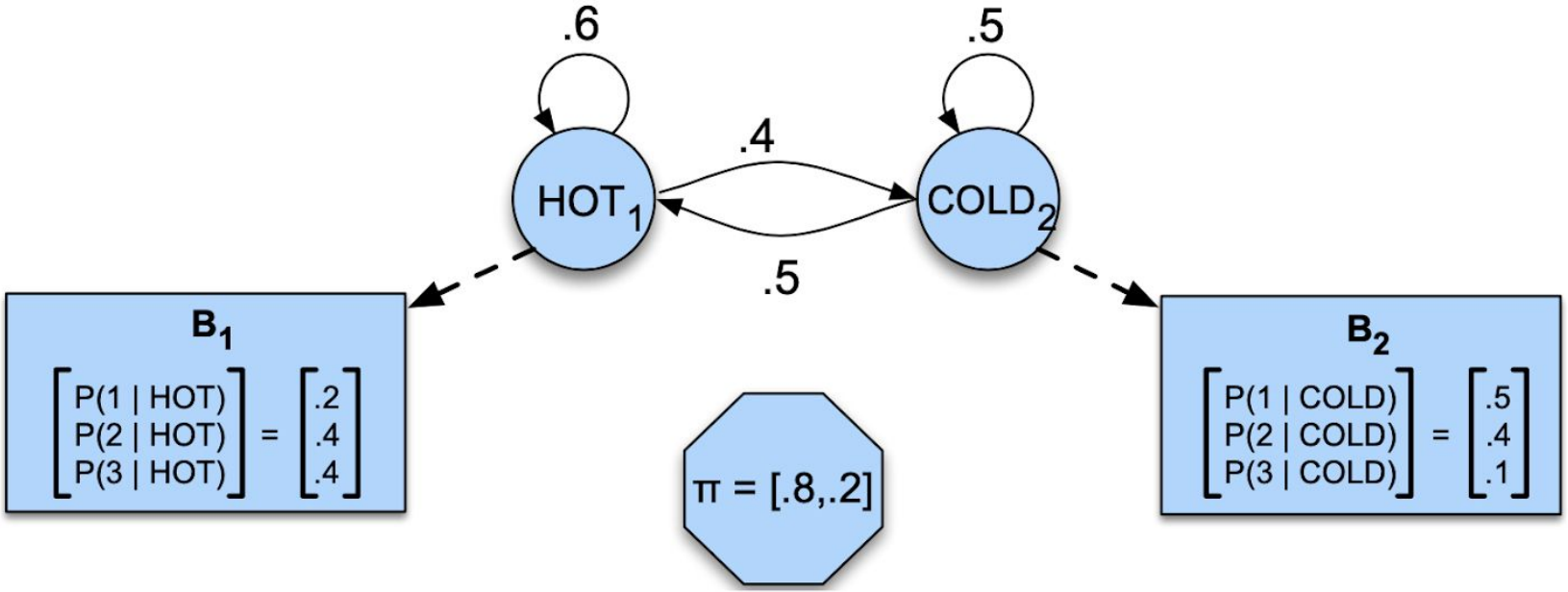
Forward Algorithm

Initialization: $\alpha_1(i) = \pi_i b_i(o_1) \quad 1 \le i \le N$

Induction: $\alpha_t(j) = \left[\sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij} \right] b_j(o_t) \quad 2 \le t \le T, \quad 1 \le j \le N$

Termination: $P(O \mid \lambda) = \sum_{i=1}^{N} \alpha_T(i)$

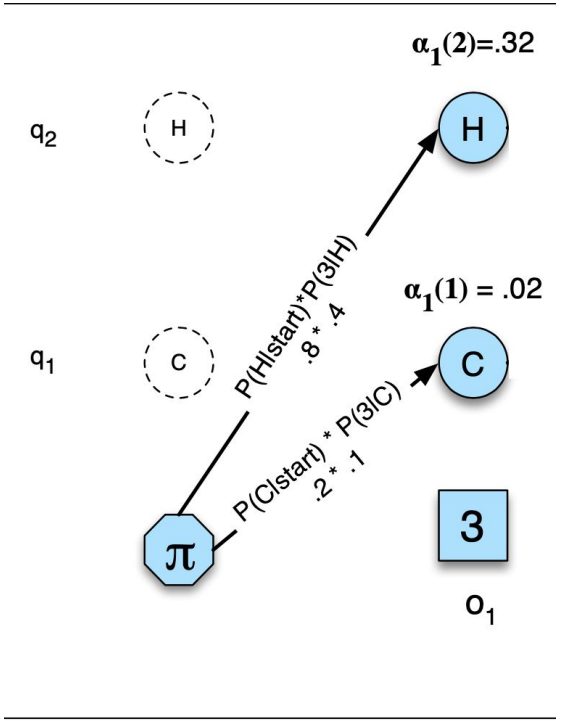
Ice Cream HMM with CPTs



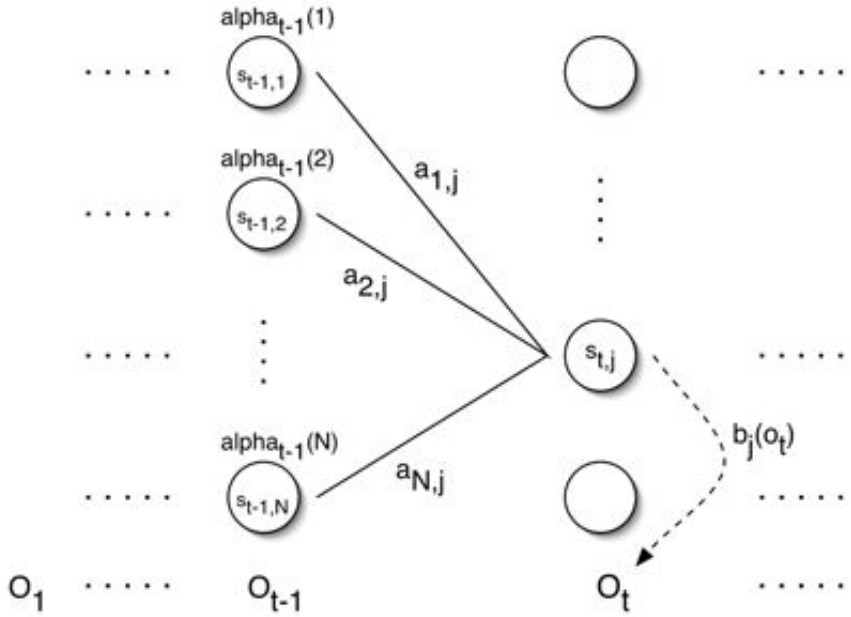
Ice Cream Example: 3 1 3

- Initialization:

$$\alpha_1(i) = \pi_i b_i(o_1) \quad 1 \le i \le N$$



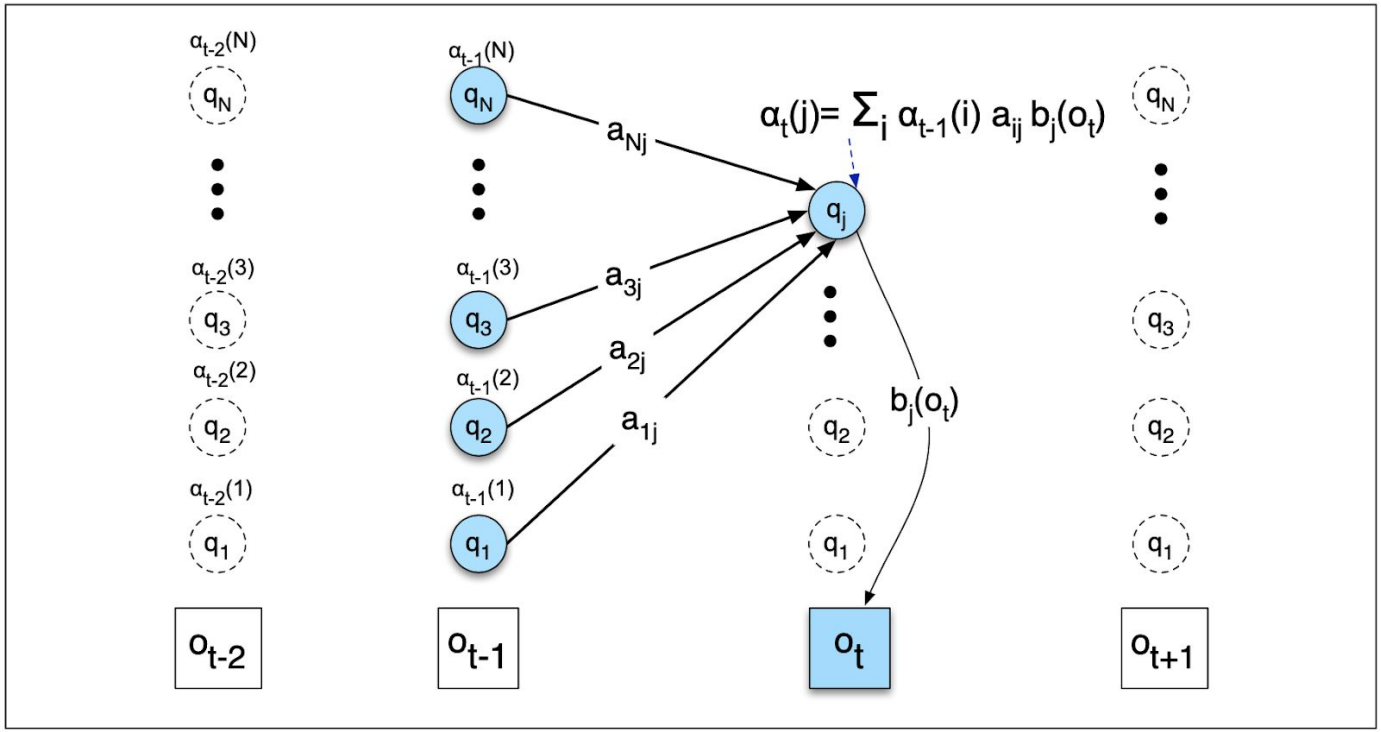
Induction Step



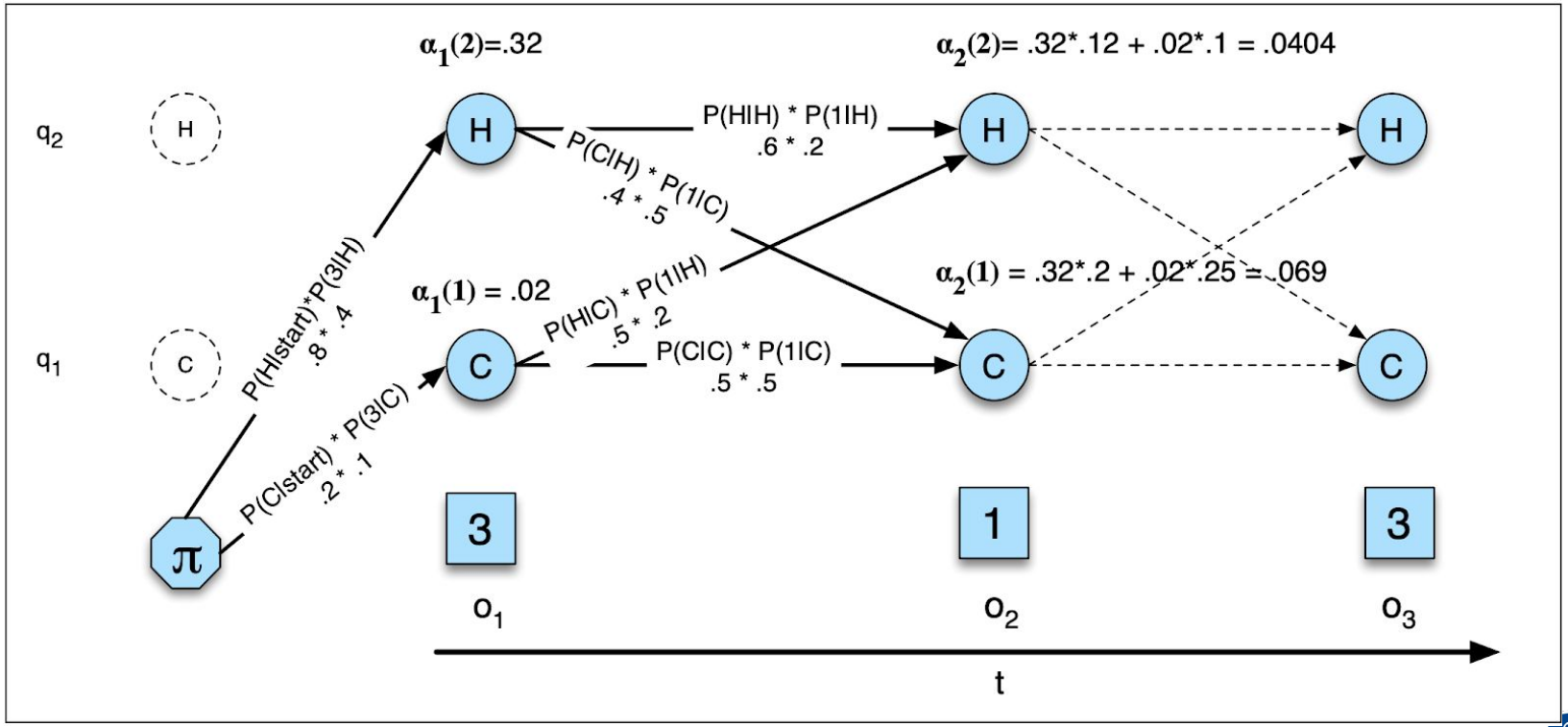
$$\alpha_t(j) = \left[\sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij} \right] b_j(o_t)$$

Forward Algorithm

Visualizing Computation of a Single Trellis Entry



Trellis for an Ice Cream Sequence: 3,1,3



Forward Algorithm

Forward Algorithm: Pseudo Code

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

create a probability matrix *forward[N,T]*
for each state s **from** 1 **to** N **do** ; initialization step
$forward[s,1] \leftarrow \pi_s * b_s(o_1)$
for each time step t **from** 2 **to** T **do** ; recursion step
for each state s **from** 1 **to** N **do**

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

$$forwardprob \leftarrow \sum_{s=1}^{N} forward[s,T]$$; termination step

return *forwardprob*

Hidden Markov Models

J&M Appendix A; AIMA 15

CSE 597: Natural Language Processing

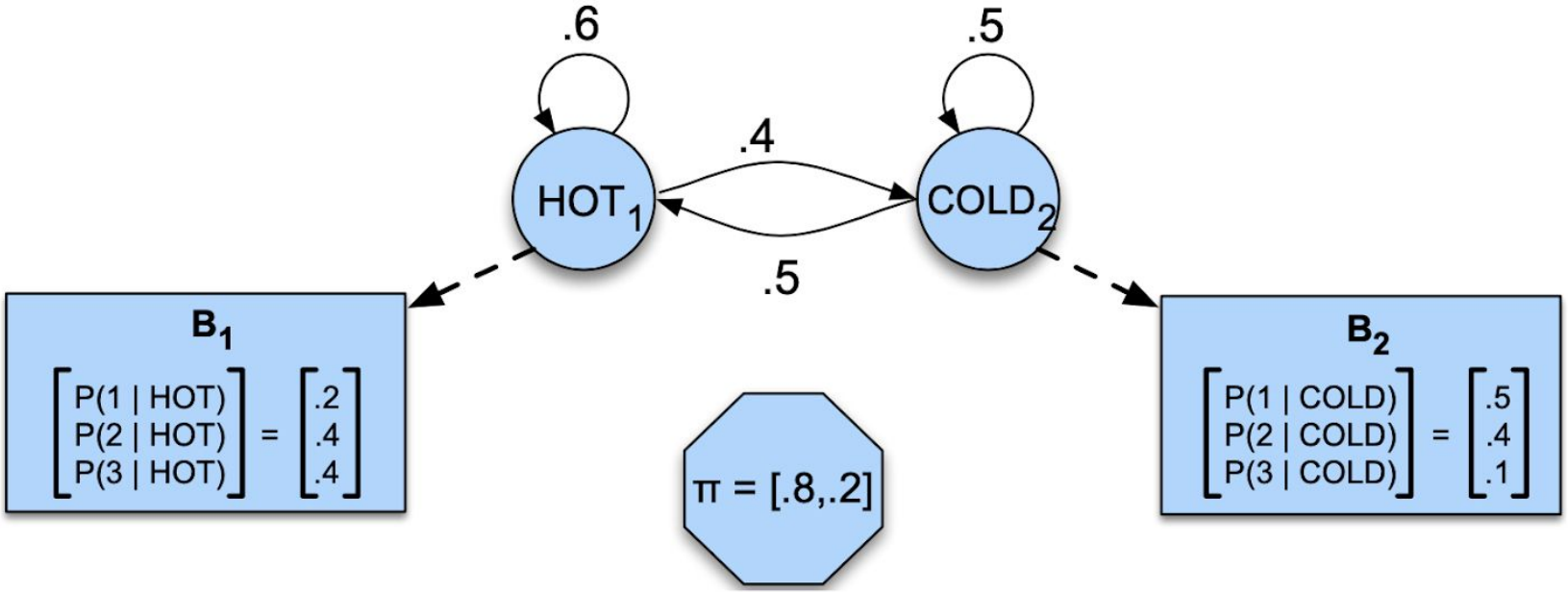
Viterbi Algorithm

Likelihood versus Decoding: Ice Cream Example

Given an HMM λ:

- Computing the **likelihood** of a sequence of observations $P(o_1,o_2,o_3)$ relies on the **forward algorithm**
  - The trellis entries carry forward all paths to each state that can emit o_i
  - The likelihood involves **summing over combinations** of state sequences that could produce a given observation sequence
- Computing the **most likely** sequence of states given the observations (decoding) $P(Q_1,Q_2,Q_3|o_1,o_2,o_3)$ relies on the Viterbi algorithm
  - The trellis entries carry forward all paths to each state that can emit o_i
  - Decoding finds the single **maximum probability** state sequence

Likelihood versus Decoding: Ice Cream Example



Decoding in Brief

- Given as input an HMM λ = (A,B) and a sequence of observations $O=o_1, o_2, ..., o_T$, find the most probable sequence of states $Q=q_1 q_2 ..., q_T$
- What is the single **most likely sequence of states** that generated $o_1=3, o_2=1, o_3=3$?

$$Q = \arg\max_{Q'} P(Q' \mid O, \lambda)$$

Forward Recursion versus Viterbi Recursion

- Forward recursion

$$\alpha_t(j) = \left[\sum_{i=1}^{N} \alpha_{t-1}(i)\, a_{ij} \right] b_j(o_t)$$

- Viterbi recursion

$$\delta_t(j) = \left[\max_{1 \le i \le N} \delta_{t-1}(i)\, a_{ij} \right] b_j(o_t)$$

Viterbi Recursion

$v_{t-1}(i)$	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

Figure from Martin & Jurafsky, 3rd Ed., Appendix A

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

- Viterbi recursion computes the maximum probability path to state j at time t given the observation $o_1, \ldots, o_t$

Back Tracing

- Viterbi recursion computes the **maximum probability path** to state j at time T given the observation $o_1, \ldots, o_T$
- Viterbi must also identify the complete **single path** that gives this maximum probability
  - Keep backpointers
  - Find $\arg\max_j \delta_T(j)$
  - Trace backpointers from state j at time T to find the state sequence from T back to 1

Viterbi Algorithm

- Initialization ($t=1$): $\delta_1(i) = \pi_i b_j(o_1) \quad 1 \le i \le N$
- Induction and backtracing ($2 \le t \le T$):

$$\delta_t(j) = \left[\max_{1 \le i \le N} \delta_{t-1}(i)\, a_{ij} \right] b_j(o_t) \qquad \psi_t(j) = \left[\arg\max_{1 \le i \le N} \delta_{t-1}(i)\, a_{ij} \right]$$

- Termination: $q_T^* = \arg\max_{1 \le i \le N} \delta_T(i)$
- Backpointer path:

$$q_t^* = \psi_{t+1}(q_{t+1}^*) \quad t = T-1, \ldots, 1$$

Viterbi Trellis for Ice Cream Example: Induction Step

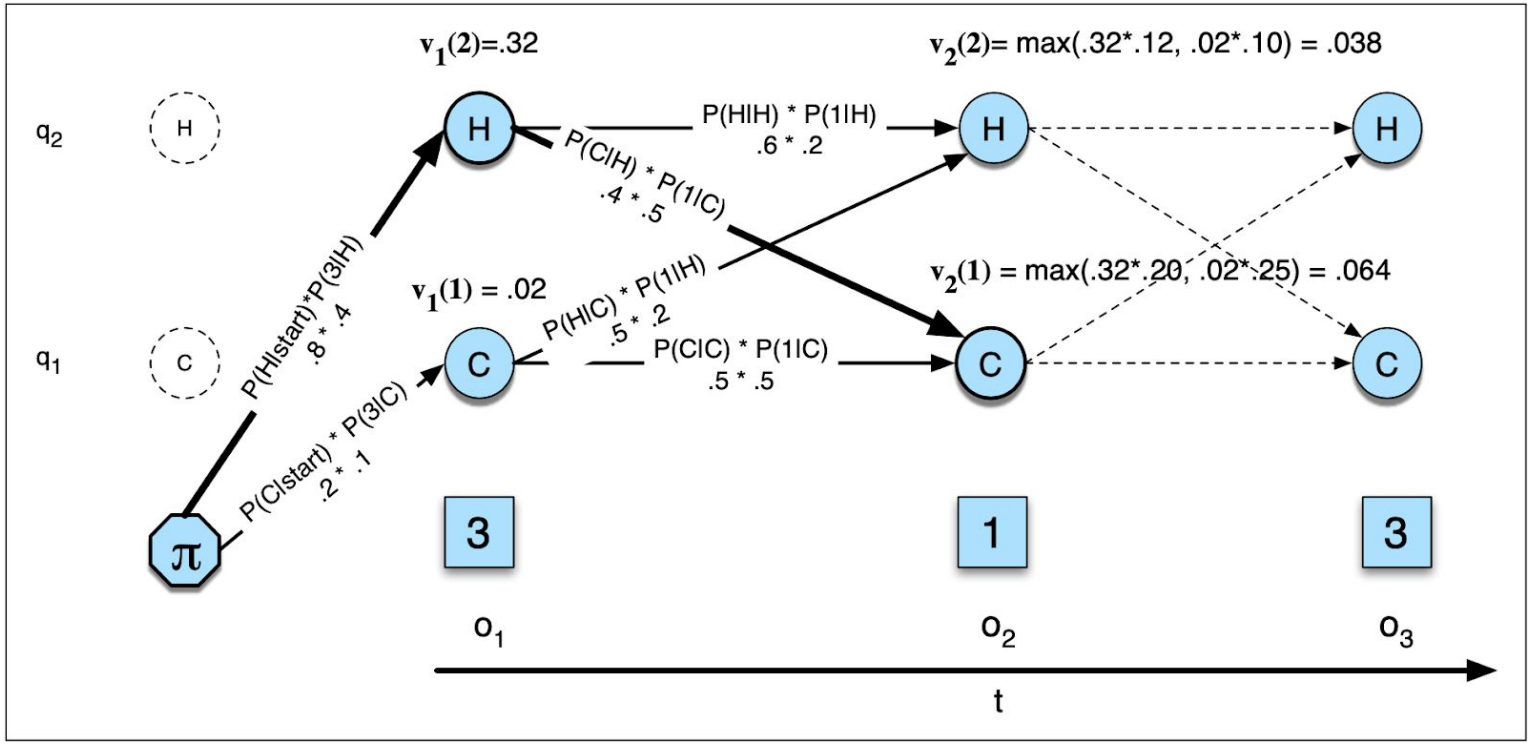


Figure from Martin & Jurafsky, 3rd Ed., Appendix A

Viterbi Trellis for Ice Cream Example: Backtracing

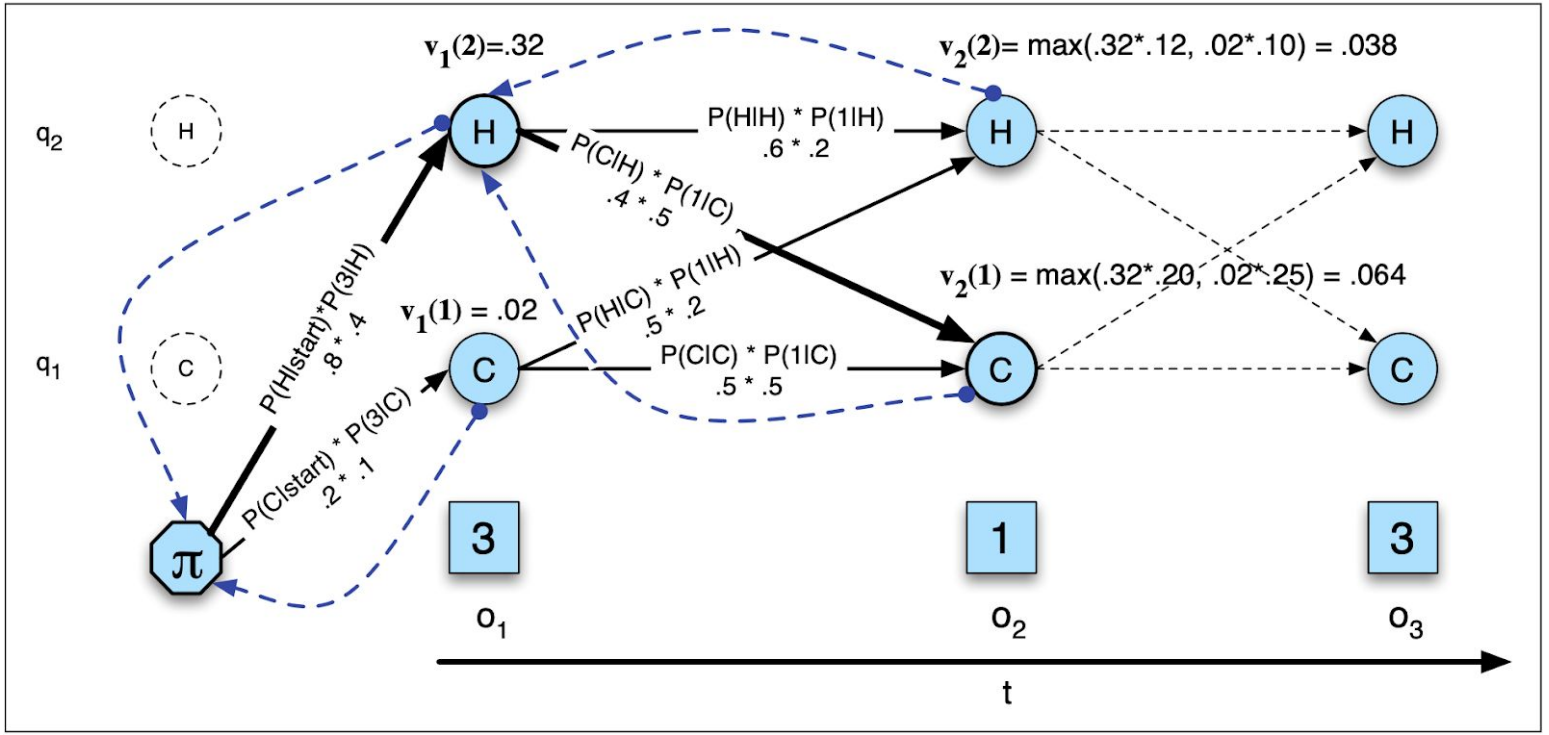


Figure from Martin & Jurafsky, 3rd Ed., Appendix A

Viterbi Pseudo Code

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

create a path probability matrix *viterbi[N,T]*
for each state *s* **from** 1 **to** *N* **do** ; initialization step
  $viterbi[s,1] \leftarrow \pi_s * b_s(o_1)$
  $backpointer[s,1] \leftarrow 0$
for each time step *t* **from** 2 **to** *T* **do** ; recursion step
  for each state *s* **from** 1 **to** *N* **do**
    $viterbi[s,t] \leftarrow \max_{s'=1}^{N} viterbi[s',t-1] * a_{s',s} * b_s(o_t)$
    $backpointer[s,t] \leftarrow \operatorname{argmax}_{s'=1}^{N} viterbi[s',t-1] * a_{s',s} * b_s(o_t)$
$bestpathprob \leftarrow \max_{s=1}^{N} viterbi[s,T]$; termination step
$bestpathpointer \leftarrow \operatorname{argmax}_{s=1}^{N} viterbi[s,T]$; termination step
bestpath ← the path starting at state *bestpathpointer*, that follows backpointer[] to states back in time
return *bestpath*, *bestpathprob*

Figure from Martin & Jurafsky, 3rd Ed., Appendix A

Hidden Markov Models

J&M Appendix A; AIMA 15

CSE 597: Natural Language Processing

POS Tagging

Parts of Speech

- Parts of speech: traditional grammatical categories like “noun, ”“verb,” “adjective,” “adverb” . . . (and many more)
- Origins in an ancient Greek monographs on grammar and parts of speech (c. 100 B.C., Dionysius Thrax, Tryphon)

- Functions:
  - Help distinguish different word meanings: N(oun) vs. V(erb)
    - EG: *river bank* (N) vs. *she banks (V) at a credit union*
    - *EG: a bear (N) will go after honey* vs. *bear (V) with me*
  - Preprocessing for ***many*** other natural language processing tasks

POS Tagging

Pervasive in Natural Language Applications

- Machine Translation
  - Translations of nouns and verbs have different forms (prefixes, suffixes)
- Speech synthesis
  - “Close” as an adjective versus verb
  - see table
- Sense disambiguation of word meanings
  - *“Close”* (adjective) as in near something
  - *“Close”* (verb) as in shut something

Noun	Verb
IN’ sult	in SULT’
OB’ ject	ob JECT’
OVER’ flow	over FLOW’
DIS’ count	dis COUNT’
CON’ tent	con TENT’

Majority Class Baseline for POS Tagging

- Assign the most frequent tag
  - 83.3% correct here
  - Typically 90% correct

Typical for NLP: a large percentage of cases are **very easy**; rest are **very hard**

Power law distribution of language phenomena, and the 80/20 rule:80% of the data comes from 20% of (common) cases

Word	POS listing in Brown Corpus		
heat	Noun (86)	***Verb*** (4)	
oil	***Noun*** (87)		
in	***prep*** (18,975)	noun (1)	adv-particle (4)
a	***det*** (21,872)	noun (1)	
large	***adj*** (347)	noun (2)	adv (5)
pot	***noun*** (28)		

POS Tagsets: Penn TreeBank

Penn Treebank, developed from 1989-1996, and very widely used ever since (treebank ≡ a bank of syntactic trees, or other linguistic annotations)

- The first large annotated corpus used for training machine learned models for natural language processing (NLP) tasks; 7M words total
- 7M words of part-of-speech tagged text
  - |POS| ≅ 36 - 48 (12 for punctuation and other symbols)
- 3M words of text with syntactic parses
- 2M words of text annotated with predicate-argument structure
- 1.6M words of transcripts of spoken language annotated for disfluency

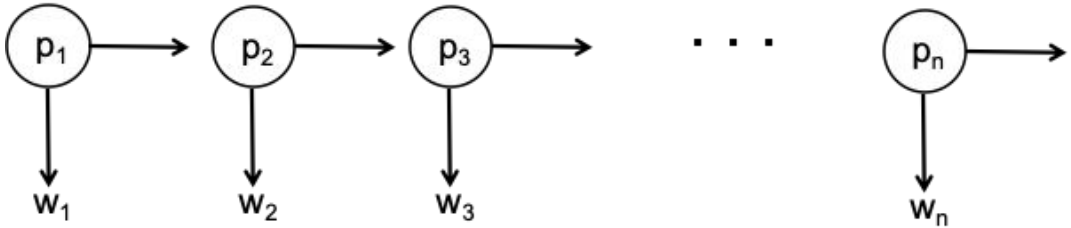
Table Showing 9 of 36 Penn TreeBank POS Tags

Tag	Description	Example
IN	preposition/subordinating conjunction	in, of, like
JJ	adjective	green
JJR	adjective, comparative	greener
JJS	adjective, superlative	greenest
MD	modal	could, will
NN	noun, singular or mass	table
NNS	noun plural	tables
NNP	proper noun, singular	John
NNPS	proper noun, plural	Vikings
VBD	verb, past tense	took
VBZ	verb, 3rd person sing. present	takes

POS Tags: Hidden States, Inherently Sequential

- Words are **observed**; part-of-speech tags are **hidden**
  - Cf. observed umbrellas vs. hidden weather states in the umbrella world
  - Cf. observed ice creams vs. hidden weather states in the ice cream world
- Likely POS sequences
  - JJ NN (delicious food, large pot)
  - NNS VBD (people voted, planes landed)
- Unlikely POS sequences
  - NN JJ (food delicious, pot large)
  - NNS VBZ (people votes, planes lands)

HMM POS Tagger as a Bayesian Network



- Condition the **hidden** POS tag p at time t on the POS tag at time $t-1$: $P(p_i|p_{i-1})$
- Condition the word w at time t on the POS tag at time t: $P(w_i|p_i)$

Summary - One

- A Hidden Markov Model relies on two Markov assumptions, one for each hidden state conditioned on the past hidden state (transition model), and one for the observations conditioned on the current hidden state (sensor model)
- Dynamic programming can reduce the computational complexity of HMMs because many computations are reused
- The trellis is the data structure used for the forward algorithm and Viterbi to support dynamic programming

Summary - Two

- The forward algorithm computes the likelihood of a sequence of observations conditioned on the sequences of states that would result in the observation sequence
- Viterbi computes the mostly likely sequence of hidden states, given an observation sequence