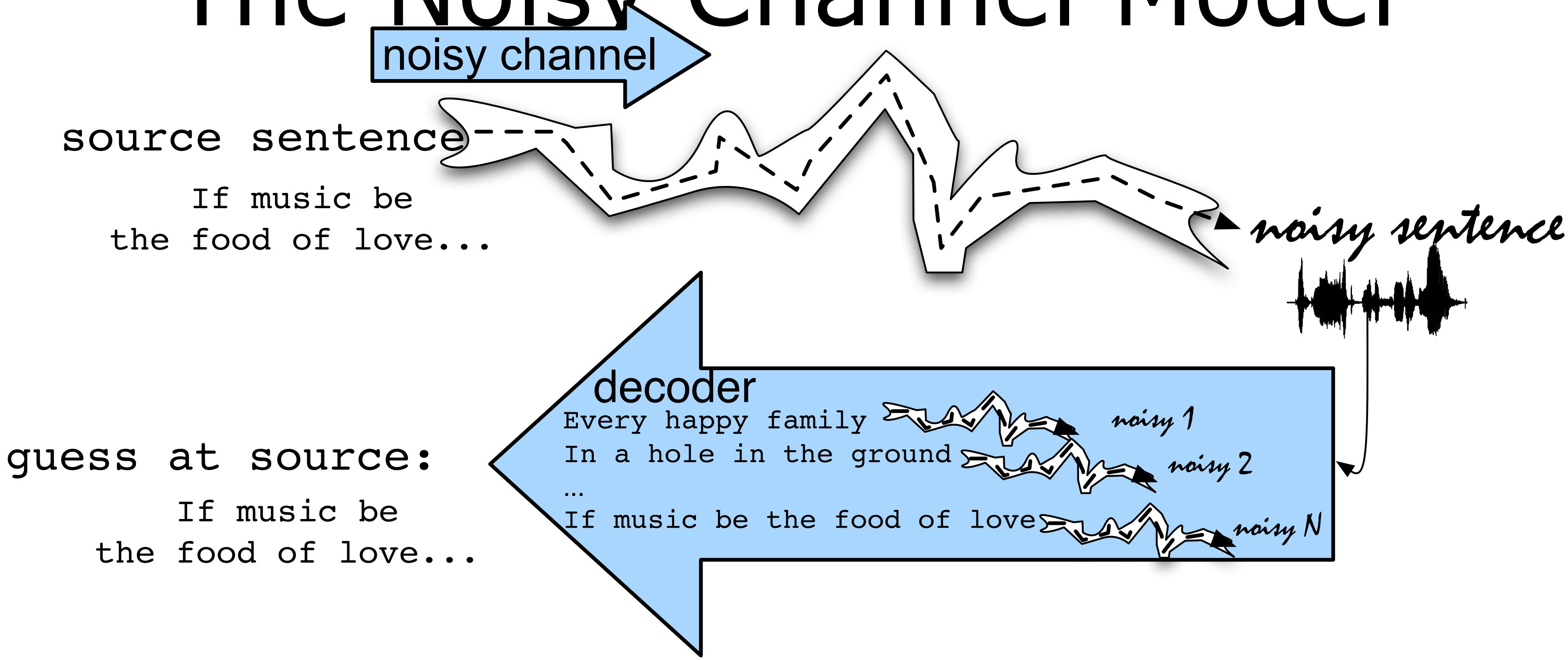
Hidden Markov Models

Sanja Lazarova-Molnar

(adapted from Dan Jurafsky's presentation)

The Noisy Channel Model



- Search through space of all possible sentences.
- Pick the one that is most probable given the waveform.

The Noisy Channel Model (II)

- What is the most likely sentence out of all sentences in the language L given some acoustic input O?
- Treat acoustic input O as sequence of individual observations
  - $O = o_1,o_2,o_3,\ldots,o_t$
- Define a sentence as a sequence of words:
  - $W = w_1,w_2,w_3,\ldots,w_n$

Noisy Channel Model (III)

- Probabilistic implication: Pick the highest prob S:

$$\hat{W} = \underset{W \in L}{\arg\max}\, P(W \mid O)$$

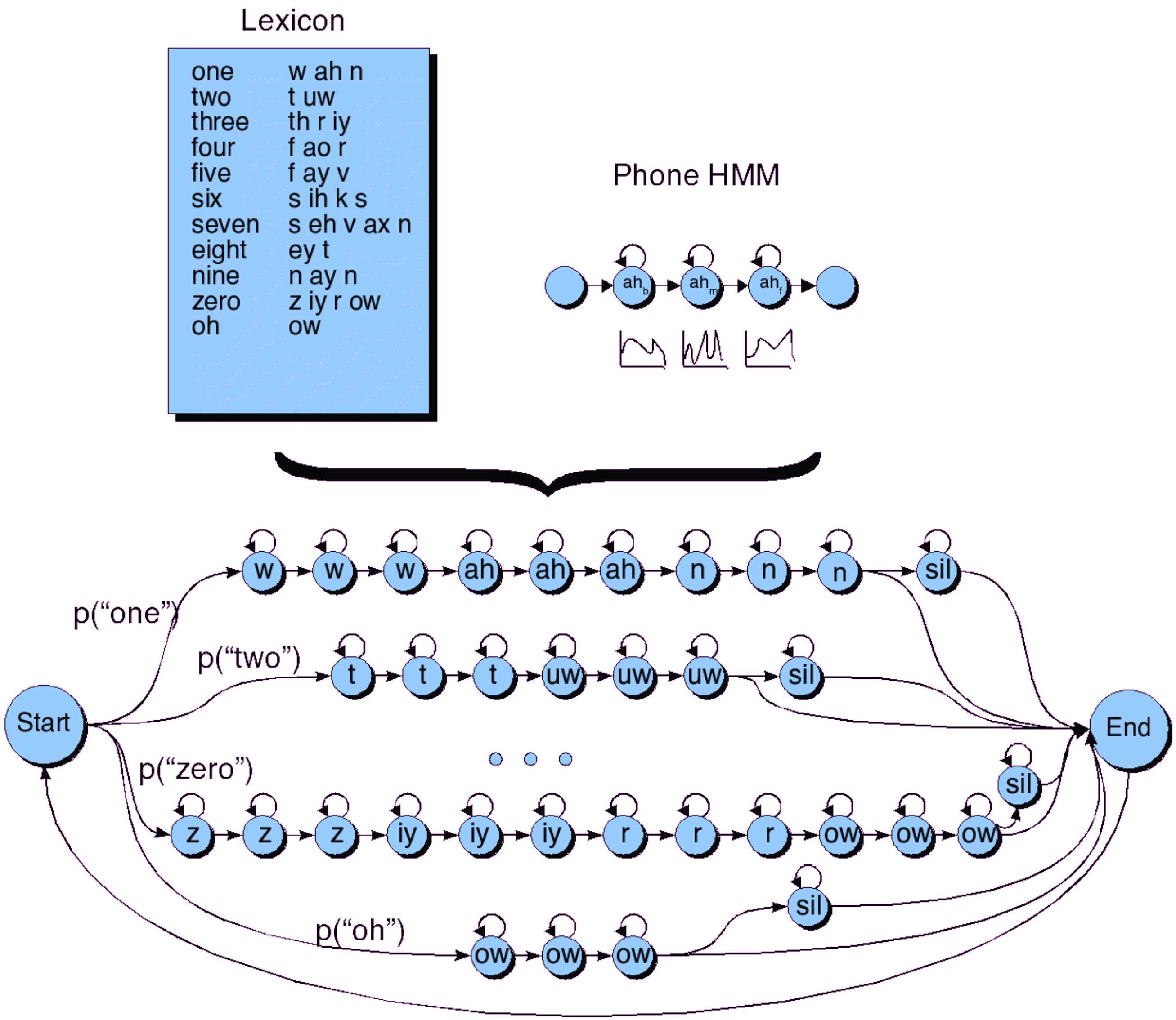
- We can use Bayes rule to rewrite this:

$$\hat{W} = \underset{W \in L}{\arg\max}\, \frac{P(O \mid W)P(W)}{P(O)}$$

- Since denominator is the same for each candidate sentence W, we can ignore it for the argmax:

$$\hat{W} = \underset{W \in L}{\arg\max}\, P(O \mid W)P(W)$$

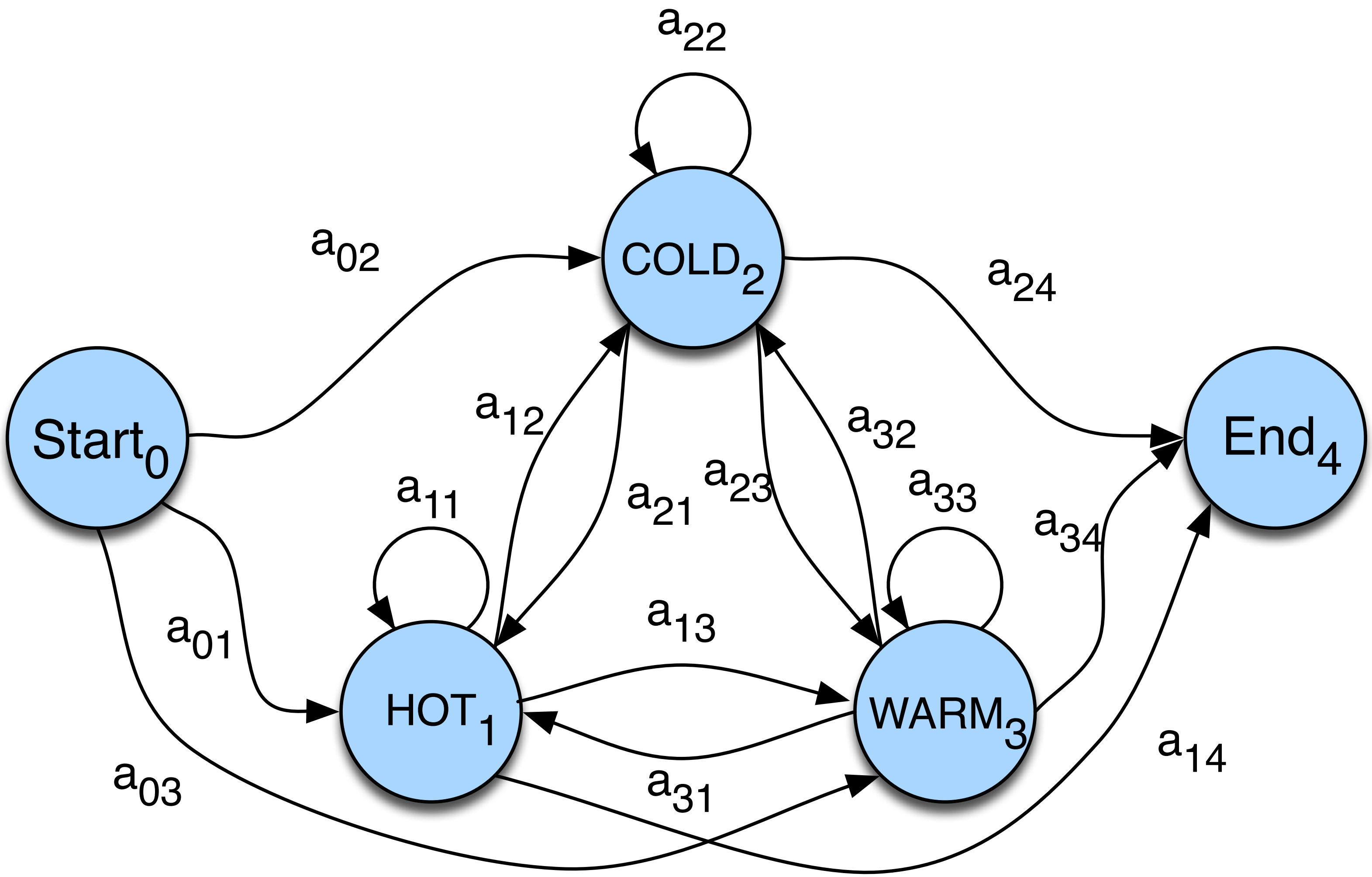
HMM for the digit recognition task



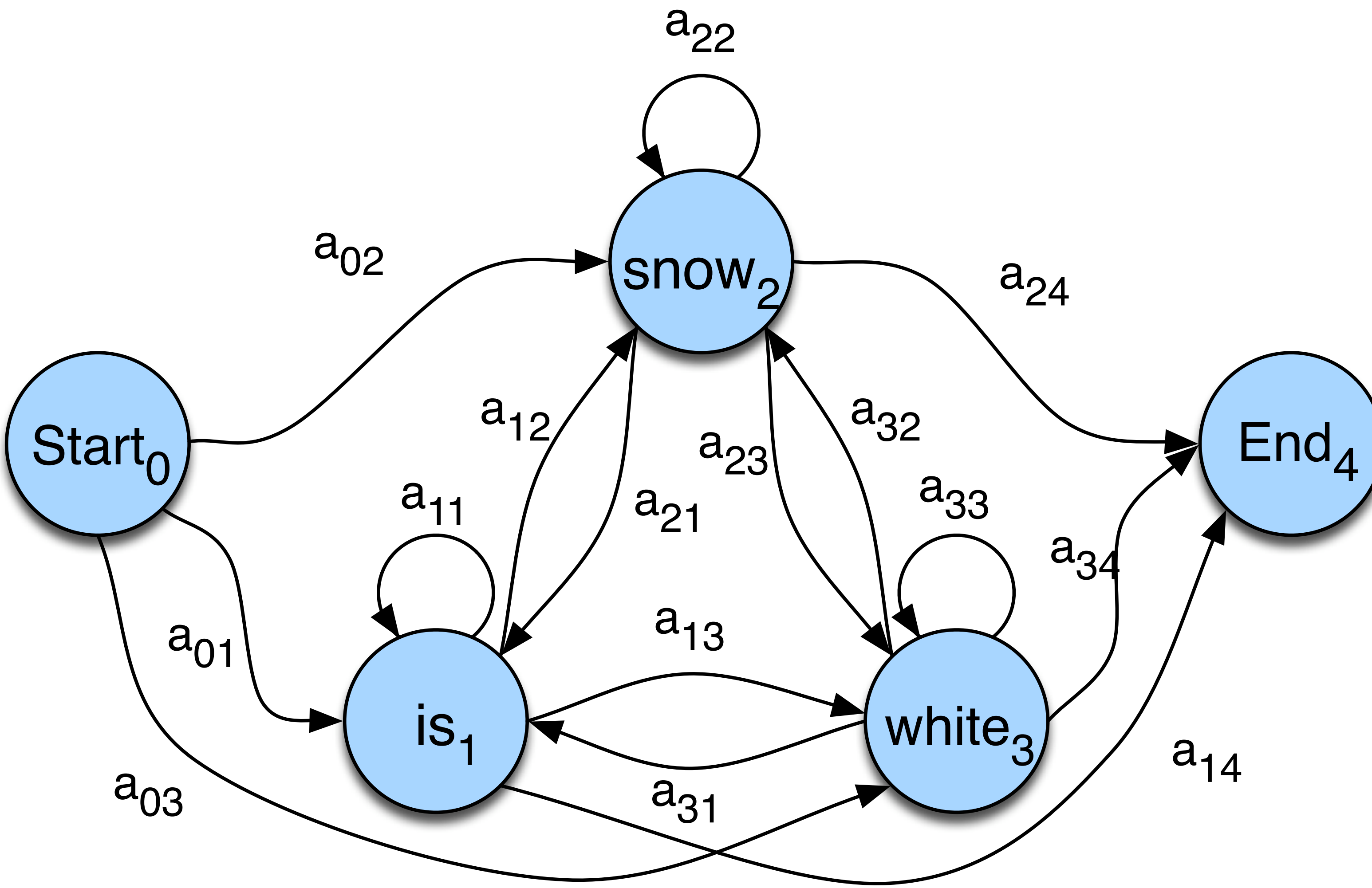
More formally: Toward HMMs

- A weighted finite-state automaton (WFSA)
  - An FSA with probabilities on the arcs
  - The sum of the probabilities leaving any arc must sum to one
- A Markov chain (or observable Markov Model)
  - a special case of a WFSA in which the input sequence uniquely determines which states the automaton will go through

Markov chain for weather



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"

- Current state only depends on previous state

$$\text{Markov Assumption}: \quad P(q_i \mid q_1 \ldots q_{i-1}) = P(q_i \mid q_{i-1})$$

Another representation for start state

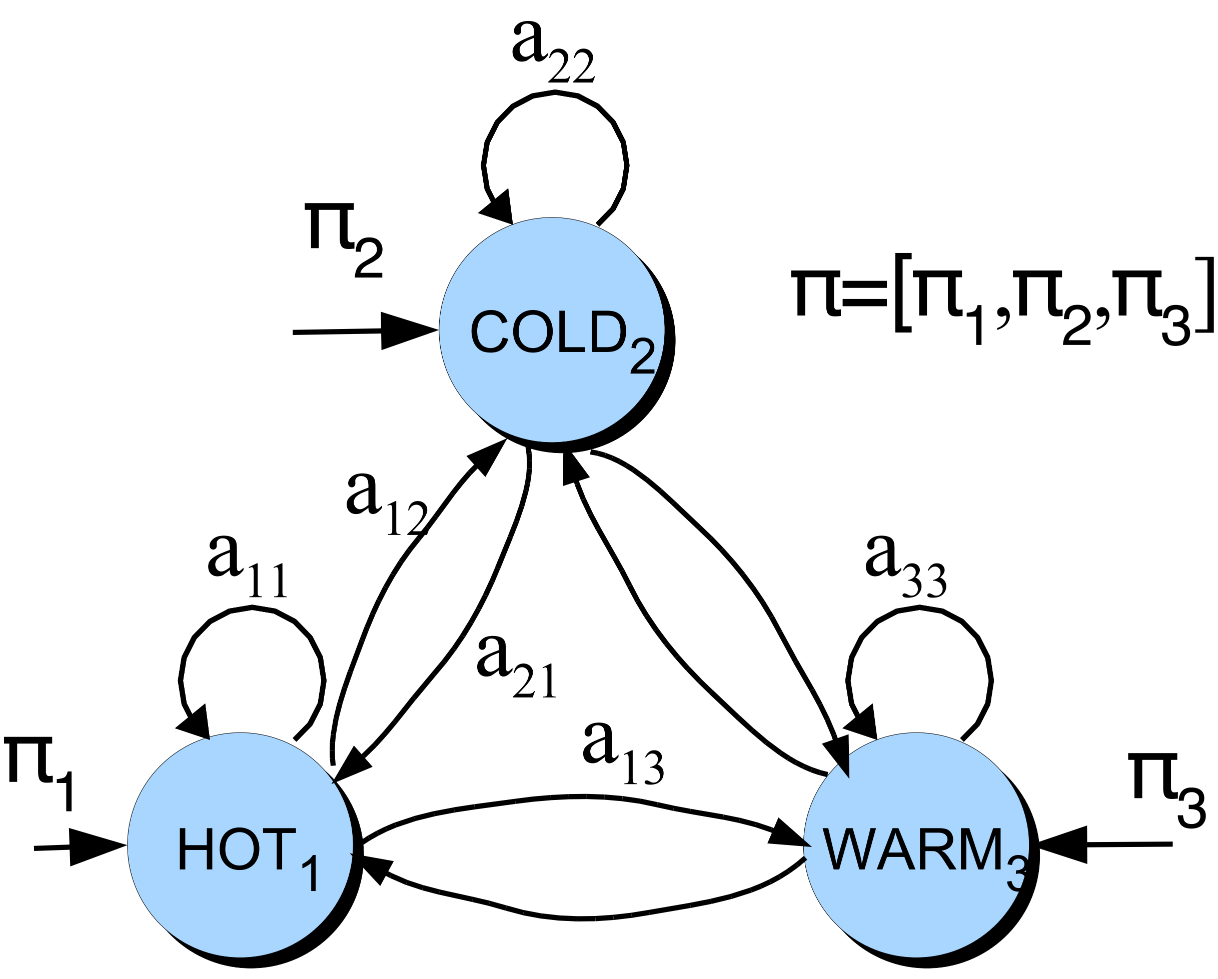
- Instead of start state
- Special initial probability vector π
  - An initial distribution over probability of start states

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

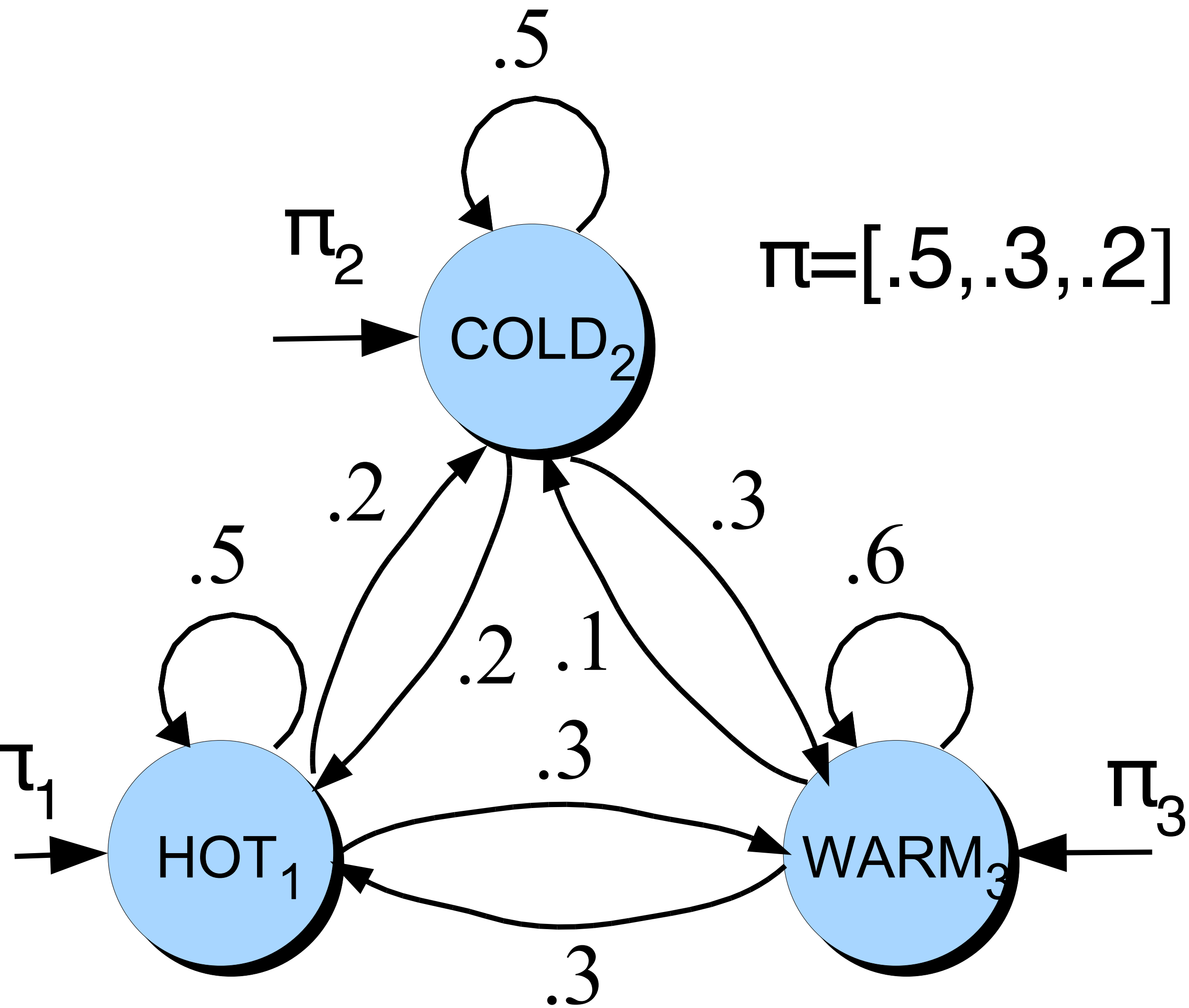
- Constraints:

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

The weather figure using pi



The weather figure: 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) =
  - $\pi_3 a_{33} a_{33} a_{33} = 0.2 \times (0.6)^3 = 0.0432$

How about?

- Hot hot hot hot
- Cold hot cold hot

- What does the difference in these probabilities tell you about the real world weather info encoded in the figure?

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 Baltimore, MD for summer of 2008
- But you find Jason Eisner's diary
- Which lists how many ice-creams Jason 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:**

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

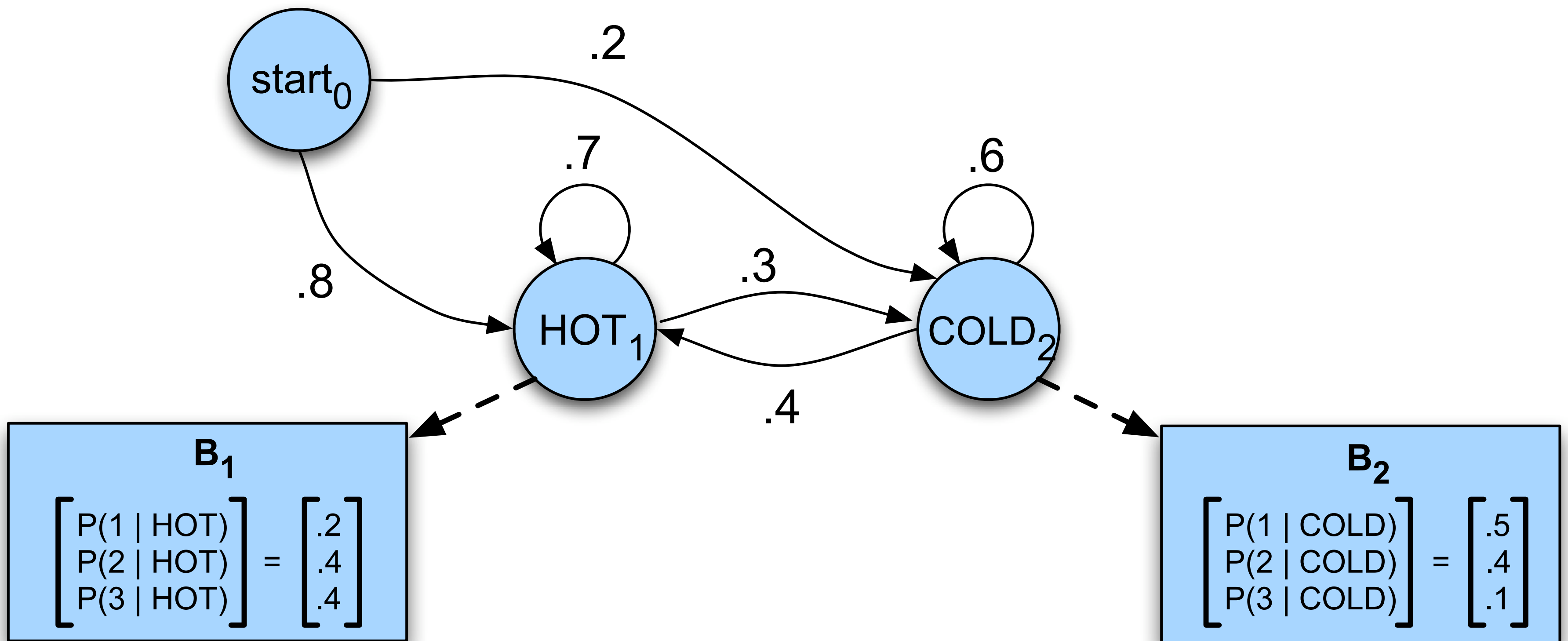
- **Output-independence assumption**

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

Eisner task

- Given
  - Ice Cream Observation Sequence: 1,2,3,2,2,2,3…
- Produce:
  - Weather Sequence: H,C,H,H,H,C…

HMM for ice cream



The Three Basic Problems for HMMs

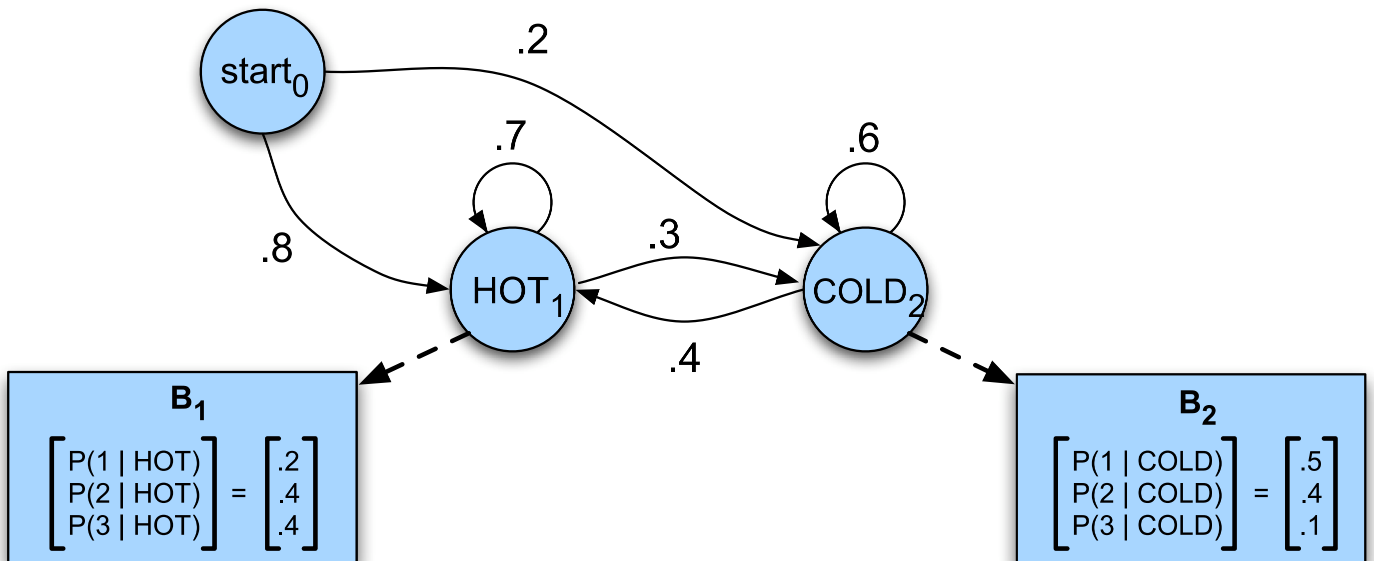
Jack Ferguson at IDA in the 1960s

- Problem 1 (**Evaluation**)**:** Given the observation sequence $O=(o_1o_2...o_T)$, 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 $O=(o_1o_2...o_T)$, and an HMM model $\Phi = (A, B)$, how do we choose a corresponding state sequence $Q=(q_1q_2...q_T)$ 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?

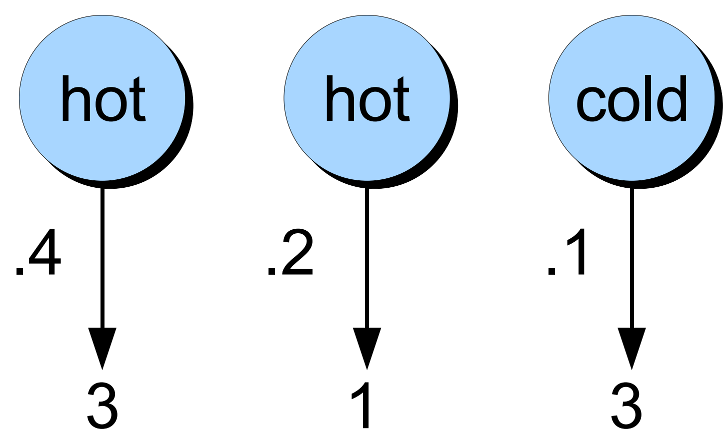
How to compute likelihood

- 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 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 Jason would eat.
  - I.e. P(3 1 3 | H H C)

Computing likelihood of 3 1 3 given hidden state sequence

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

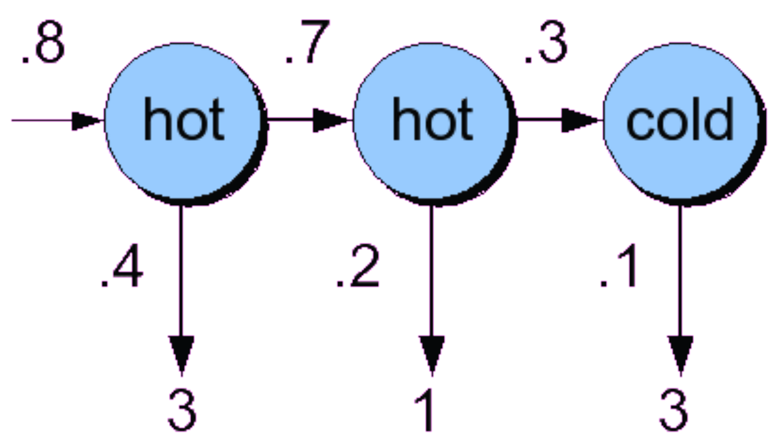
$$P(3\ 1\ 3|\text{hot hot cold}) \ = \ P(3|\text{hot}) \times P(1|\text{hot}) \times P(3|\text{cold})$$



Computing joint probability of observation and state sequence

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

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



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?

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

- 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
  - 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 probabiltv

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

The Forward Recursion

1. Initialization:

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

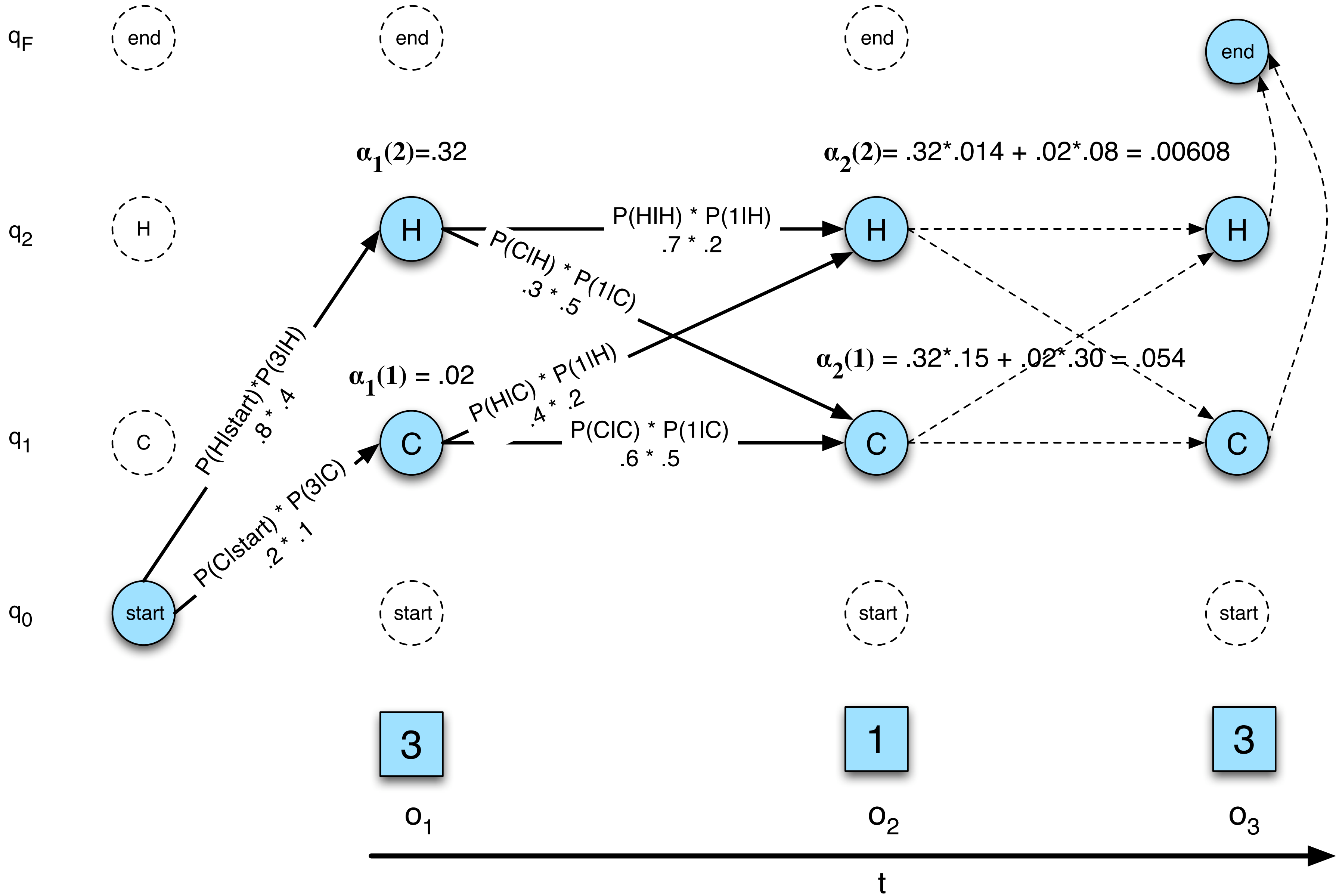
2. Recursion (since states 0 and F are non-emitting):

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

3. Termination:

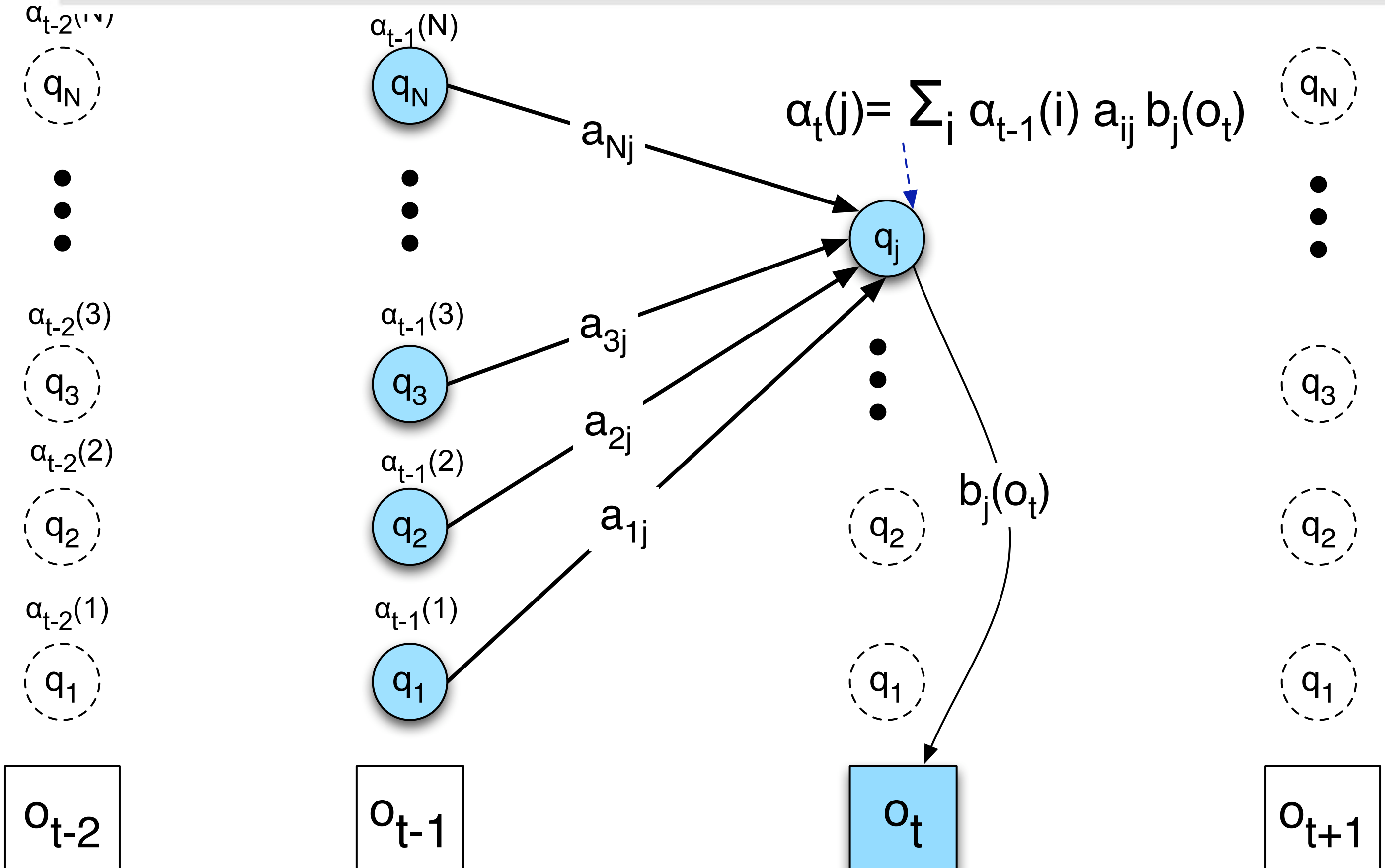
$$P(O|\lambda) = \alpha_T(q_F) = \sum_{i=1}^{N} \alpha_T(i)\, a_{iF}$$

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
$\quad forward[s,1] \leftarrow a_{0,s} * b_s(o_1)$
for each time step t **from** 2 **to** T **do** ; recursion step
$\quad$ **for** each state s **from** 1 **to** N **do**

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

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

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

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

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

Viterbi Recursion

1. **Initialization:**

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

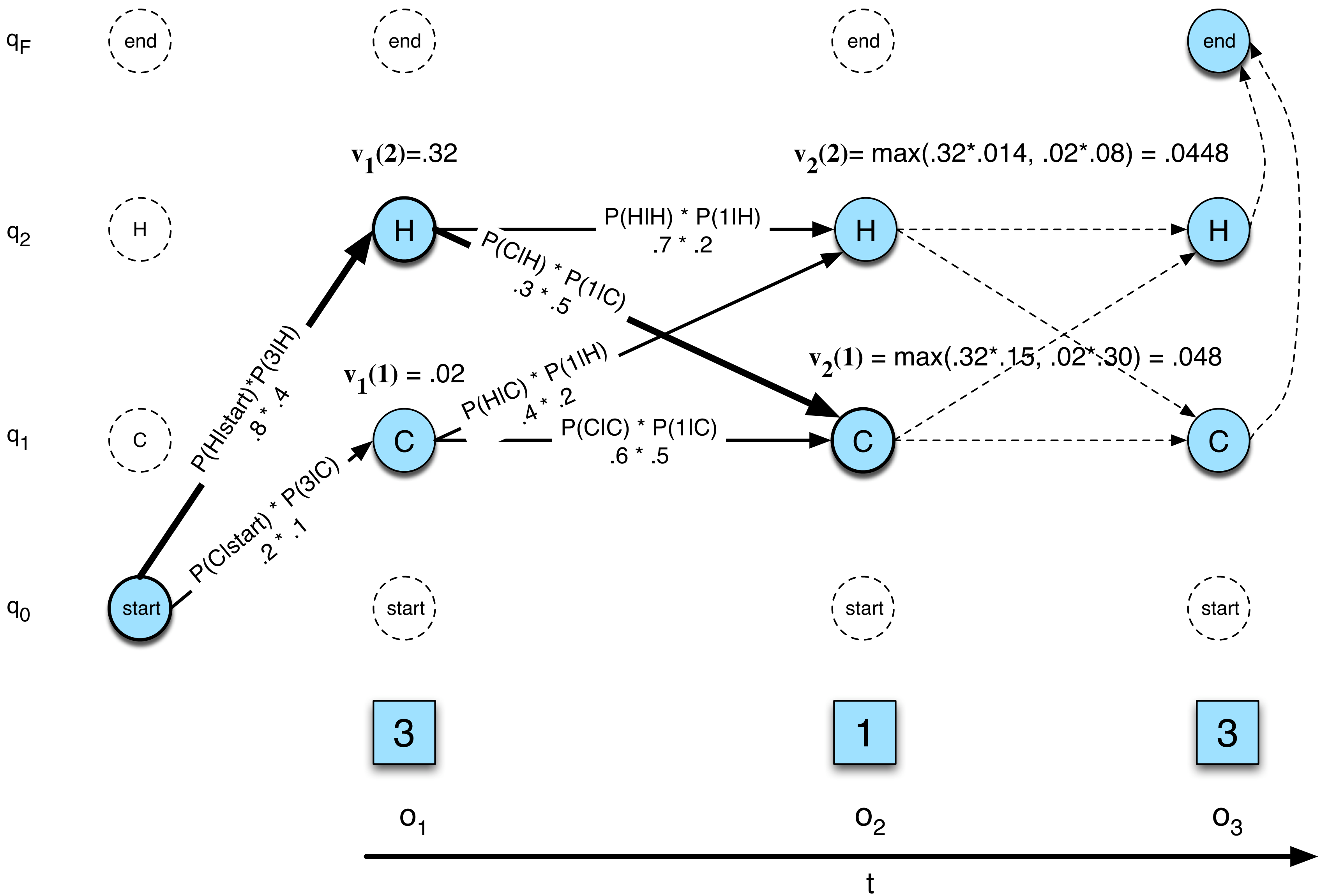
2. **Recursion** (recall that states 0 and q_F are non-emitting):

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

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