Hidden Markov Models

COSC 6336 Intro to Natural Language Processing

Spring 2018

With adapted material from Yang Liu, who borrowed material from Tanja Schultz and Dan Jurafsky

In This Lecture

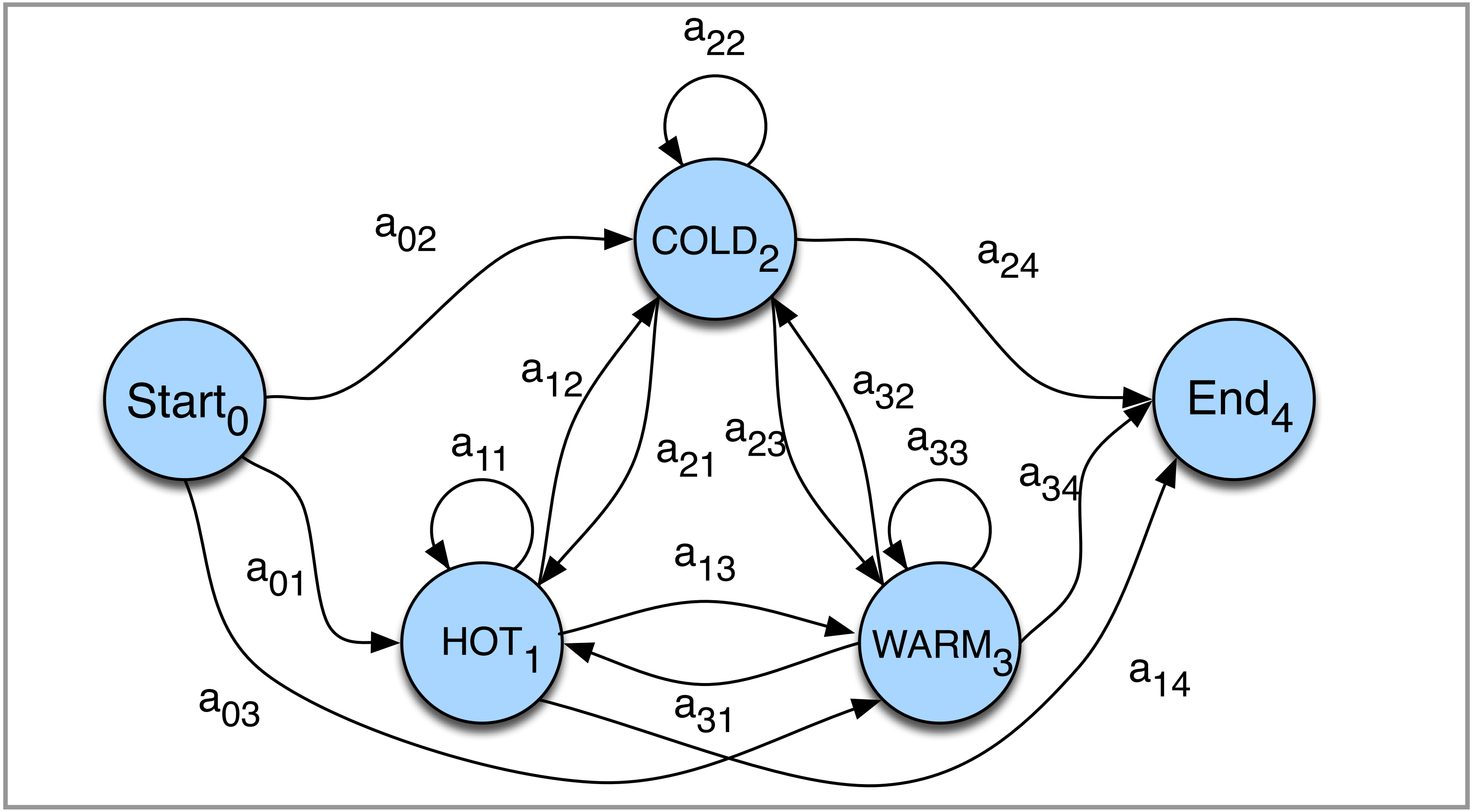
- Introduction to Hidden Markov Models (HMMs)
  - Forward algorithm
  - Viterbi algorithm

More Formally: Toward HMMs

Markov Models

- 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 chains can't represent inherently ambiguous problems
  - Useful for assigning probabilities to unambiguous sequences

Markov Chain for Weather



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
- Current state only depends on previous state

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

- Transition probability matrix A

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

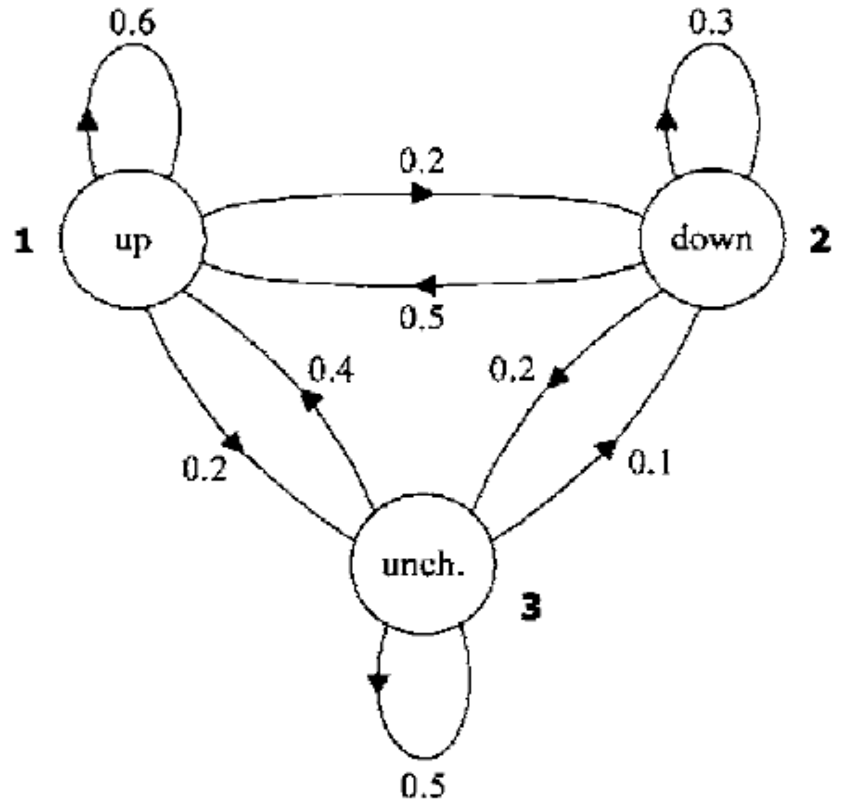
- Special initial probability vector π

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

- Constraints:

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

Markov Model for Dow Jones



Initial state probability matrix

$$\boldsymbol{\pi} = (\pi_i) = \begin{pmatrix} 0.5 \\ 0.2 \\ 0.3 \end{pmatrix}$$

State-transition probability matrix

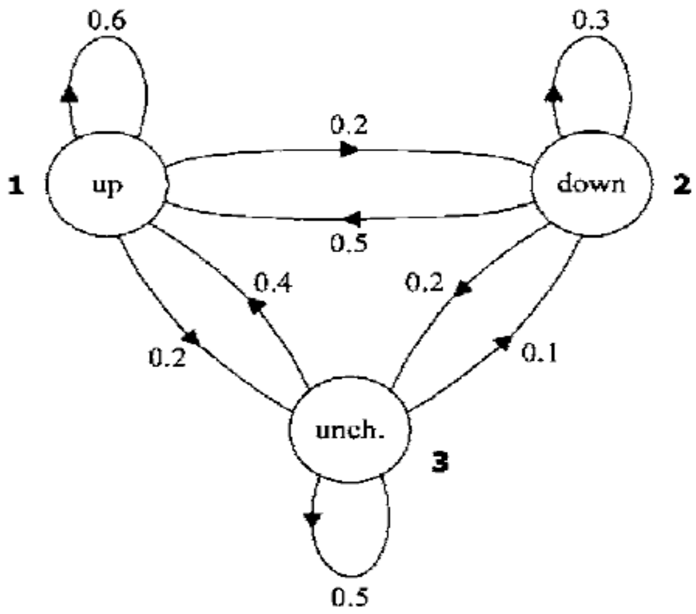
$$\mathbf{A} = \{a_{ij}\} = \begin{bmatrix} 0.6 & 0.2 & 0.2 \\ 0.5 & 0.3 & 0.2 \\ 0.4 & 0.1 & 0.5 \end{bmatrix}$$

Markov Model for Dow Jones

- What is the probability of 5 consecutive up days?
- Sequence is up-up-up-up-up
- I.e., state sequence is 1-1-1-1-1
- P(1,1,1,1,1) = ?

Markov Model for Dow Jones

- P(1,1,1,1,1) =
  - $\pi_1 a_{11} a_{11} a_{11} a_{11}$ = 0.5 x $(0.6)^4$ = 0.0648



Initial state probability matrix

$$\boldsymbol{\pi} = (\pi_i) = \begin{pmatrix} 0.5 \\ 0.2 \\ 0.3 \end{pmatrix}$$

State-transition probability matrix

$$\mathbf{A} = \{a_{ij}\} = \begin{bmatrix} 0.6 & 0.2 & 0.2 \\ 0.5 & 0.3 & 0.2 \\ 0.4 & 0.1 & 0.5 \end{bmatrix}$$

Hidden Markov Model

- For Markov chains, the output symbols are the same as the states
  - See **up** one day: we're in state **up**
- But in many NLP tasks:
  - output symbols are words
  - hidden states are something else
- 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

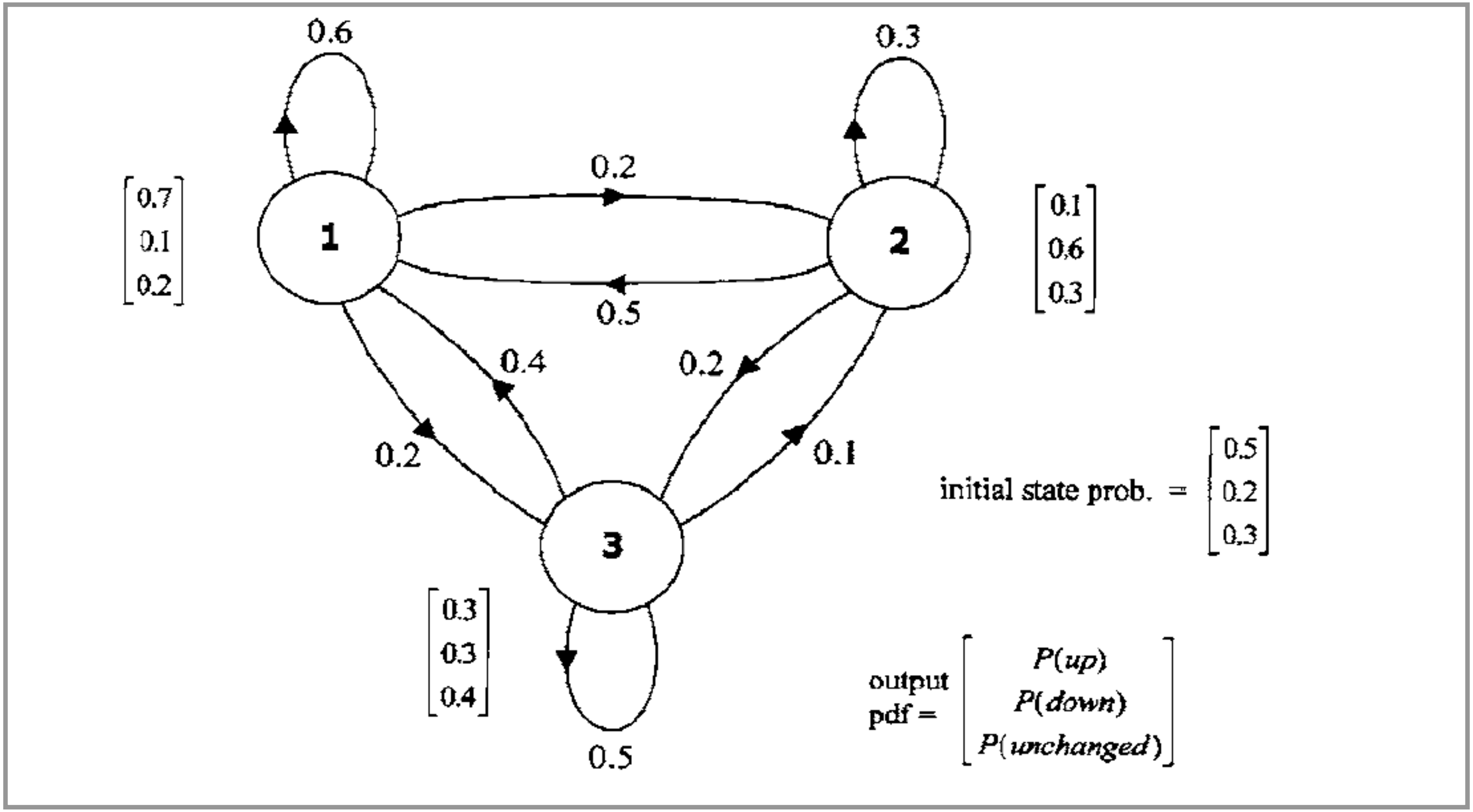
- Markov assumption:

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

- Output-independence assumption

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

HMM for Dow Jones



From Huang et al.

HMMs for Weather and Ice-cream

- Jason Eisner's cute HMM in Excel, showing Viterbi and EM:

  http://www.cs.jhu.edu/~jason/papers/#teaching

  Idea:
  - You are climatologists in 3004
  - Want to know about Baltimore weather in 2004
  - Only data you have is Jason Eisner's diary
  - Which records how much ice cream he ate each day
- Observation:
  - Number of ice creams
- Hidden State: Simplify to only 2 states
  - Weather is Hot or Cold that day.

The Three Basic Problems for HMMs

- (From the classic formulation by Larry Rabiner after Jack Ferguson)
- L. R. Rabiner. 1989. A tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. Proc IEEE 77(2), 257-286. Also in Waibel and Lee volume.

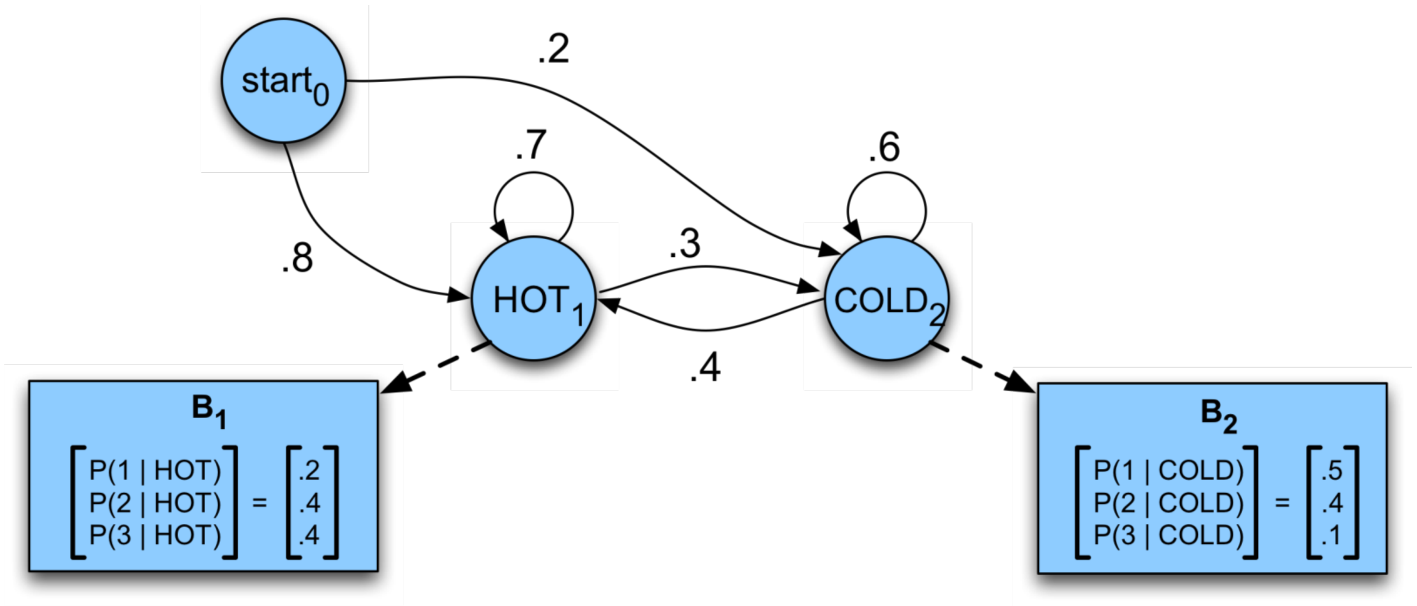
The Three Basic Problems for HMMs

- Problem 1 (**Evaluation/Likelihood**): Given the observation sequence $O=(o_1o_2\ldots o_T)$, and an HMM model Φ = (A,B), how do we efficiently compute $P(O|\ \Phi)$, the probability of the observation sequence, given Φ
- Problem 2 (**Decoding**): Given the observation sequence $O=(o_1o_2\ldots o_T)$, and an HMM model Φ = (A,B), how do we choose a corresponding state sequence $Q=(q_1q_2\ldots q_T)$ that is optimal in some sense (i.e., best explains the observations)
- Problem 3 (**Learning**): How do we adjust the model parameters Φ = (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?

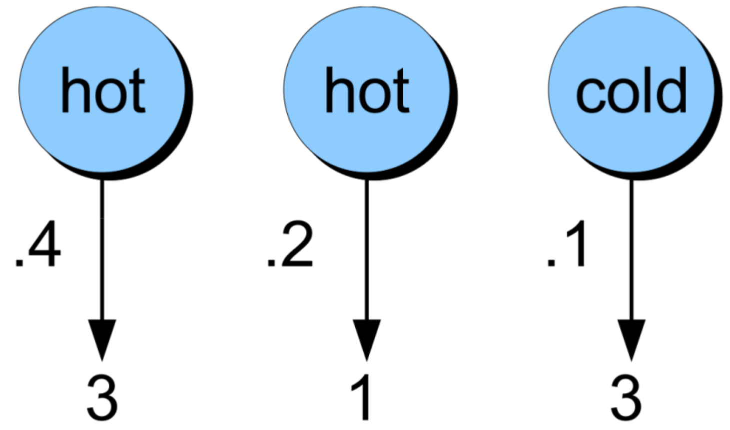
How to Compute Likelihood

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

Computing Likelihood of 3 1 3 Given Hidden State Sequence

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

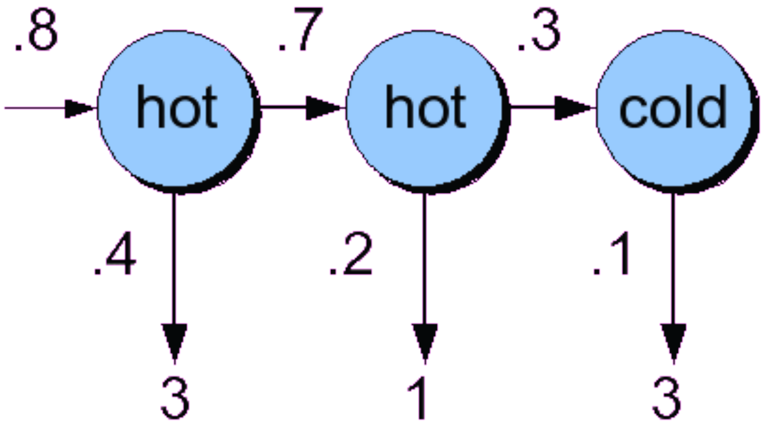
$$P(3\ 1\ 3|\text{hot hot cold}) \ = \ P(3|\text{hot}) \times P(1|\text{hot}) \times P(3|\text{cold})$$



Computing Joint Probability of Observation and a Particular 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

Computing Observation Likelihood $P(O|\Phi)$

- Why can't we do an explicit sum over all paths?
- Because it's intractable, there are $O(N^T)$ paths
- What we do instead:
- The Forward Algorithm. $O(N^2T)$
- 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

The Forward Algorithm

- The goal of the forward algorithm is to compute

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

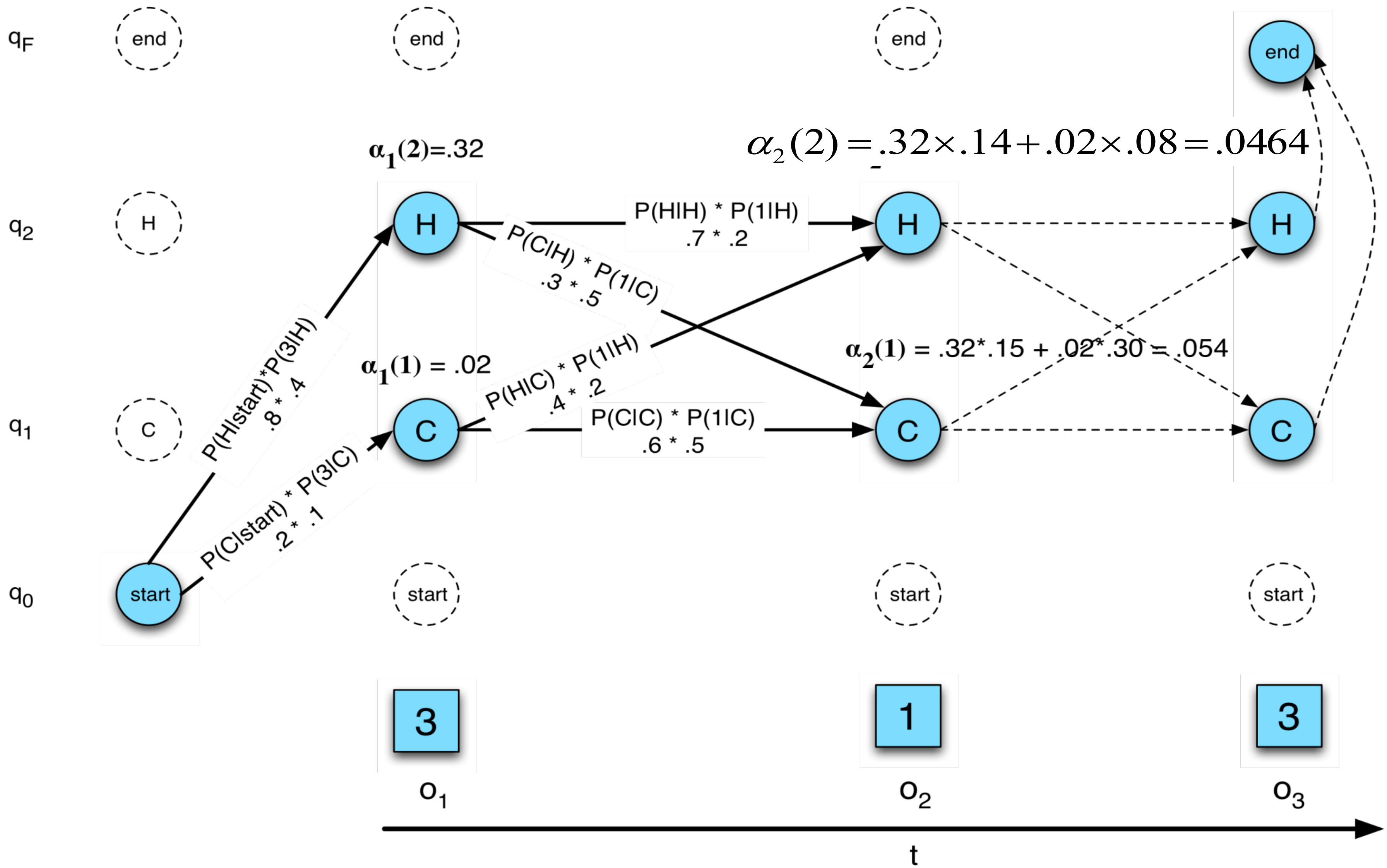
- We'll do this by recursion

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

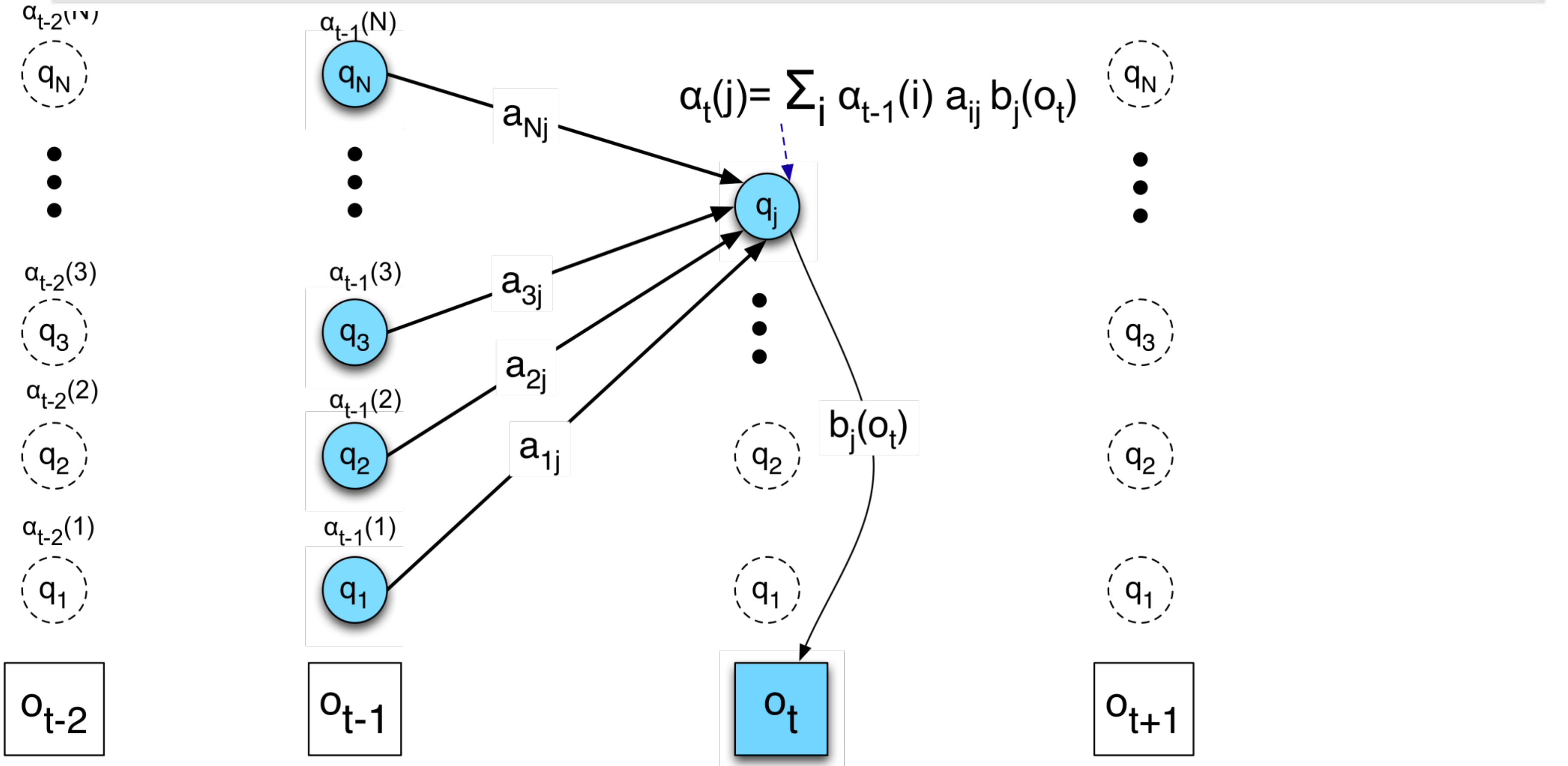
$$\alpha_t(j) = P(o_1, o_2 \ldots o_t, q_t = j | \lambda)$$

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

- The Idea: Fold these exponential paths into a simple trellis, so that all possible paths will remerge into N states at every time slice.
- We define the *forward probability* as follows: $\alpha_t(i) = P(o_0 o_1 \cdots o_t, q_t = i|\Phi)$
- this is the probability that the HMM Φ is in state i at time t having generated partial observation O_1^t.
- We compute it by induction:
  - Initialization: $\alpha_1(i) = \pi_i P(o_1|q_i), 1 \leq i \leq N$
  - (equivalently: $\alpha_1(i) = \pi_i b_i(o_1), 1 \leq i \leq N$
  - Induction:

$$\alpha_t(j) = [\sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij}] b_j(o_t), \quad 2 \leq t \leq T, 1 \leq j \leq N \tag{3}$$

  - Termination: $P(O|\Phi) = \sum_{i=1}^{N} \alpha_T(i)$

The Forward Algorithm

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

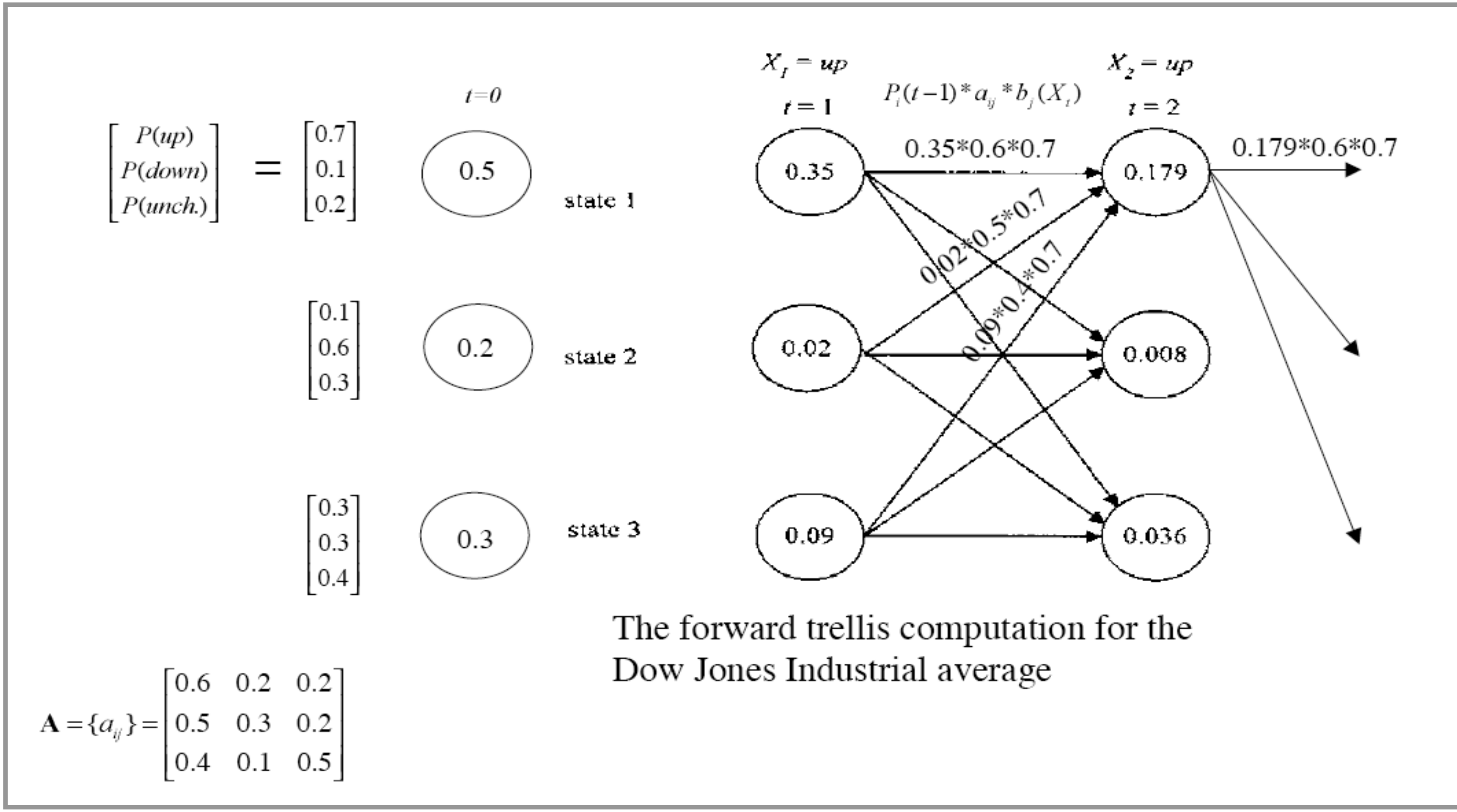
create a probability matrix *forward[N+2,T]*
for each state s **from** 1 **to** N **do** ; initialization step
$forward[s,1] \leftarrow a_{0,s} * b_s(o_1)$
for each time step t **from** 2 **to** T **do** ; recursion step
for each state s **from** 1 **to** N **do**

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

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

return $forward[q_F,T]$

Forward Trellis for Dow Jones



The forward trellis computation for the Dow Jones Industrial average

The Three Basic Problems for HMMs

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

Decoding

- Given an observation sequence
  - up up down
- And an HMM
- The task of the **decoder**
  - To find the best **hidden** state sequence
- We can calculate P(O|path) for each path
- Could find the best one
- But we can't do this, since again the number of paths is $O(N^T)$. Instead:
  - Viterbi Decoding: dynamic programming, slight modification of 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:**

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

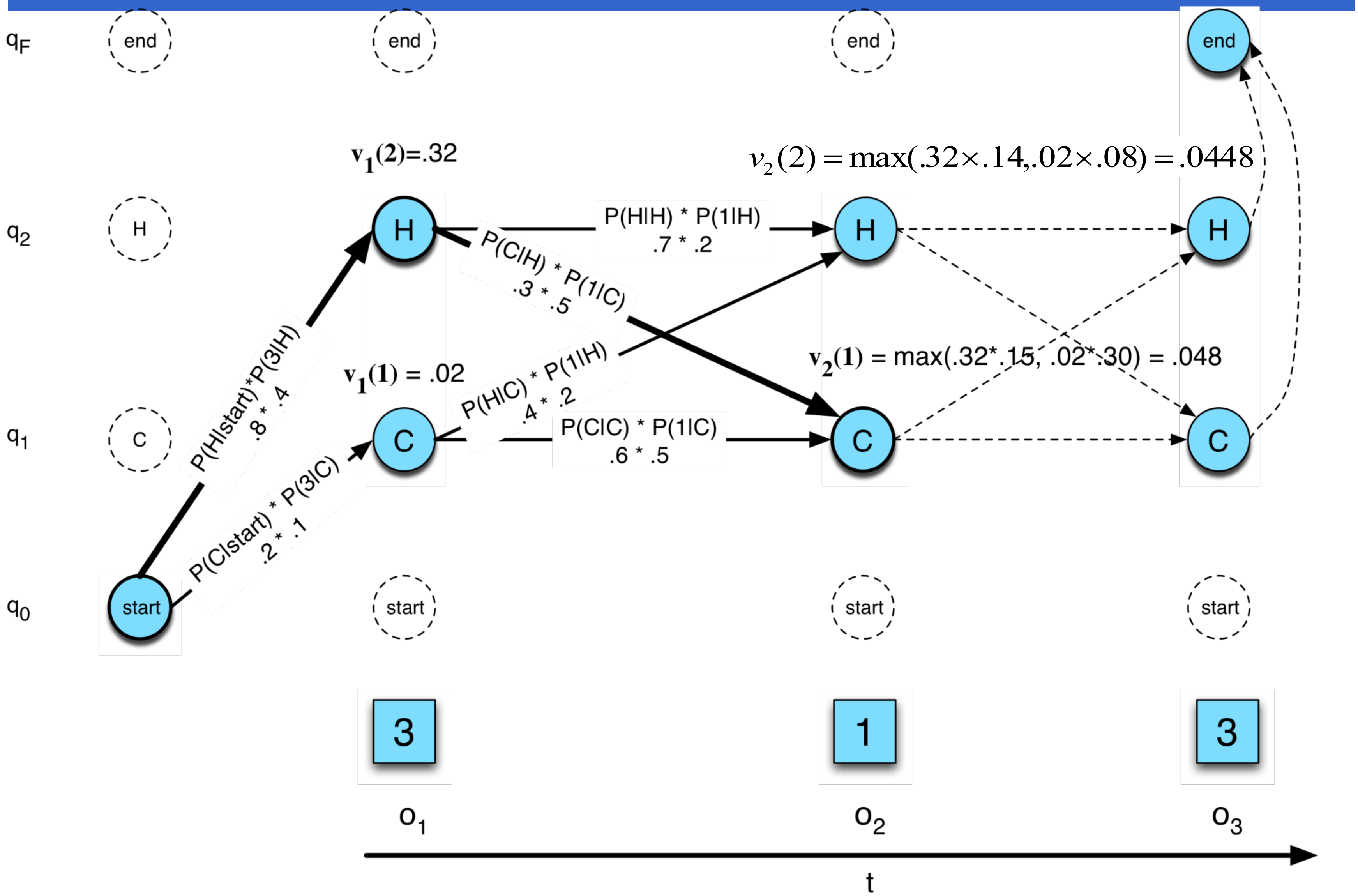
2. **Recursion** (recall that states 0 and q_F are non-emitting):

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

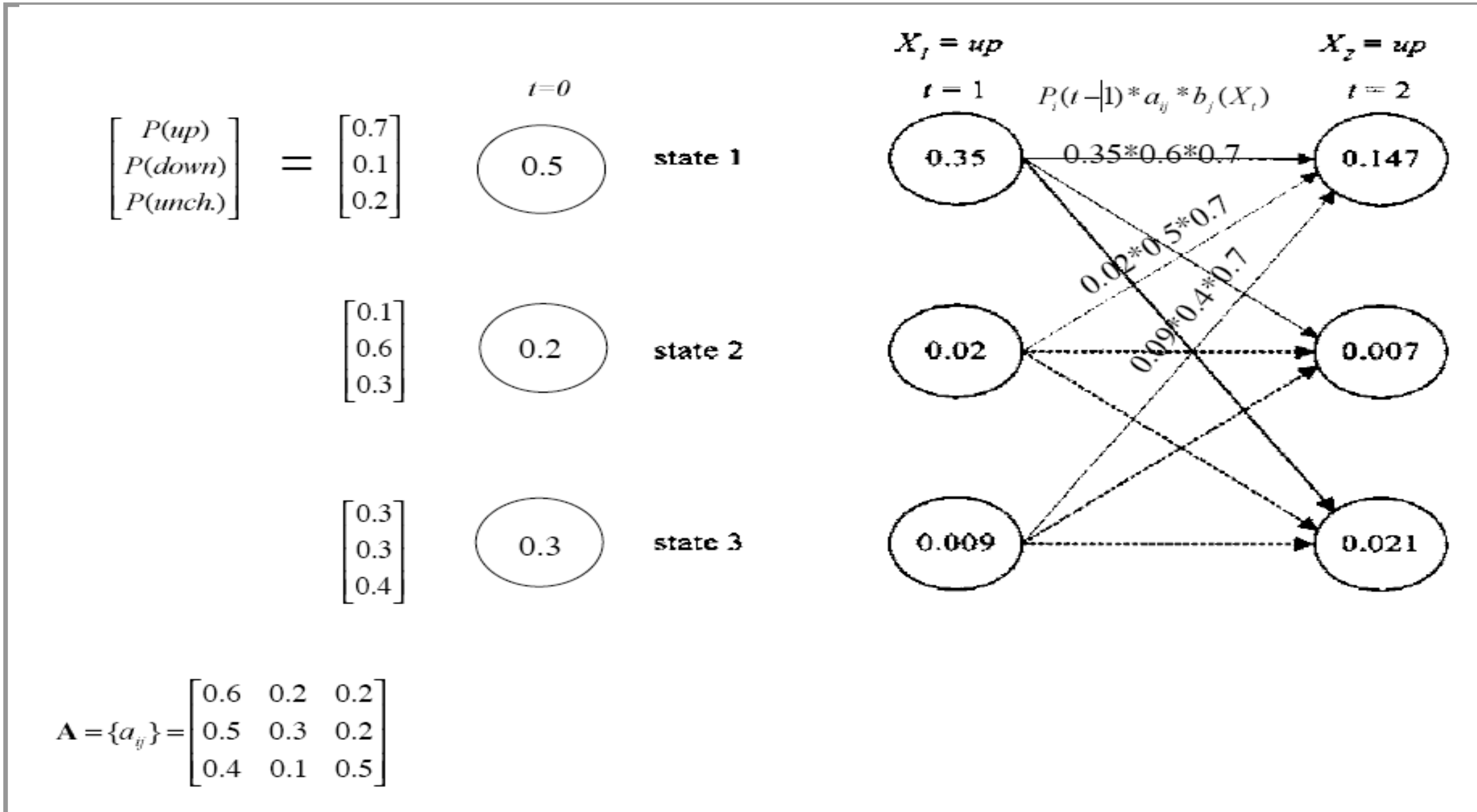
3. **Termination:**

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

The Viterbi trellis



Viterbi for Dow Jones



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

The 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 $backpointer[q_F,T]$

So Far…

- Forward algorithm for evaluation
- Viterbi algorithm for decoding
- Next topic: the learning problem