



UNIVERSITÀ
di **VERONA**

Machine Learning

HIDDEN MARKOV MODELS

Cigdem Beyan
A. Y. 2024/2025



SEQUENTIAL DATA

- Obtained through measurement of **time series data**
 - Examples:
 - Speech **recognition/classification**: Acoustic features at successive time frames
 - **Natural language processing**: sequence of characters in a sentence, successive words,...
 - Weather **forecasting**: using the data of successive days
 - Financial series **forecasting**: using daily values of currency exchange rate

TIME SERIES ANALYSIS

Financial series



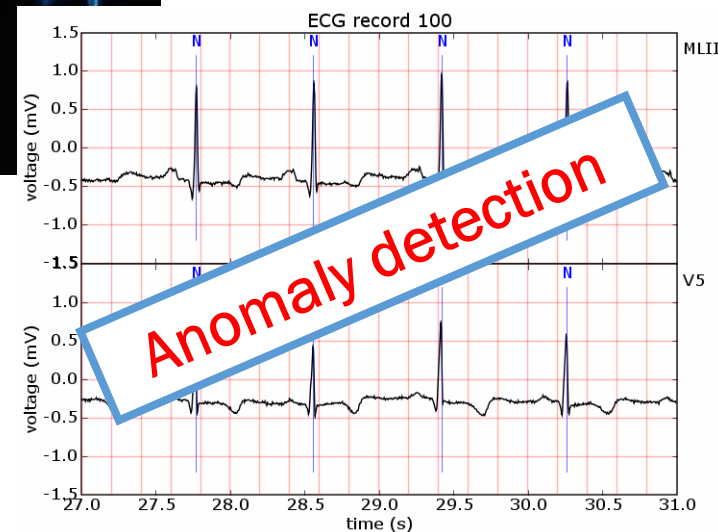
UNIVERSITÀ
di **VERONA**

Audio signals



Classification

Biomedical data

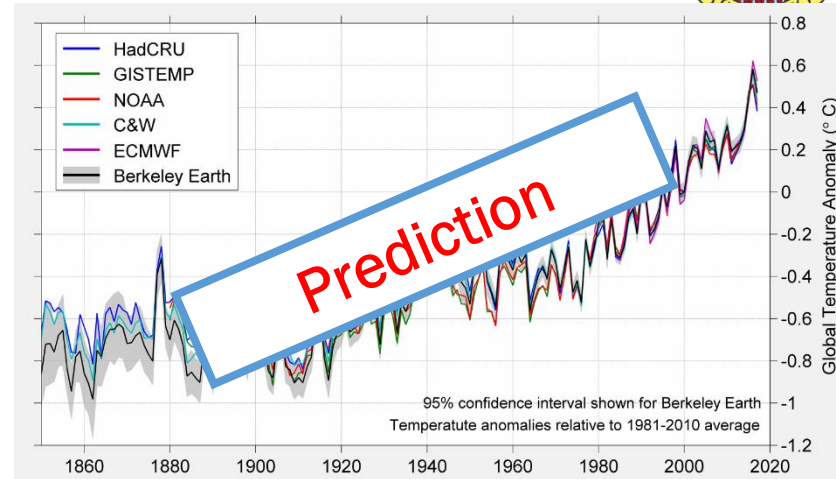


Anomaly detection

Gestures



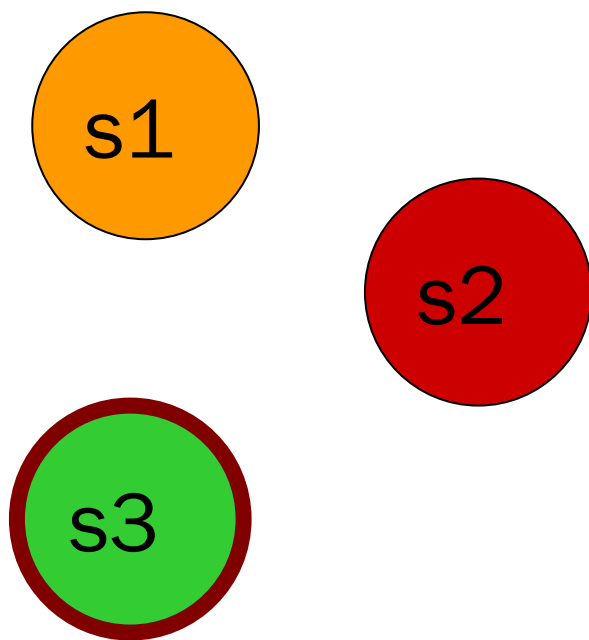
Segmentation



MARKOV MODELS (MARKOV CHAINS)

- Are **stochastic models** (i.e., incorporating randomness) describing a **sequence** of possible events in which the **probability** of each event depends only on the state attained in the **previous event**
 - Processes in time, the states at time t are influenced by a state at time $t-1$
- Aims to predict the next value from previous values, but it is impractical to consider a general dependence of future dependence on **all previous observations**
 - **Complexity** grows as the number of observations increases
- Markov models assume dependence on **the most recent observations.**

SIMPLEST MODEL BASED ON INDEPENDENCE

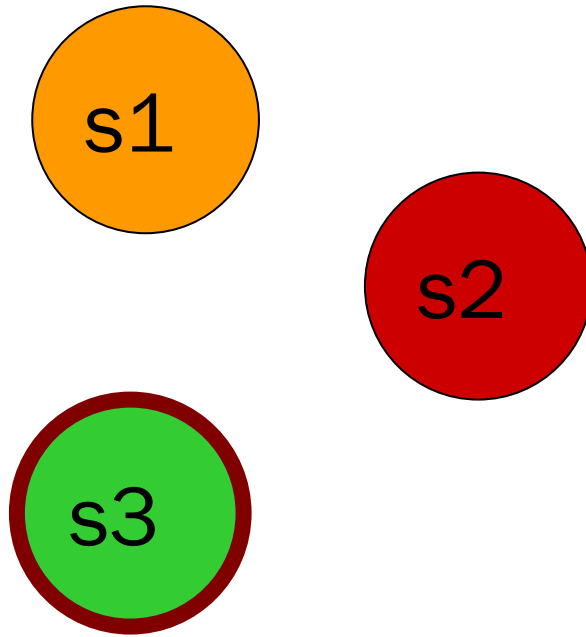


$N=3$

N states s_1, s_2, \dots, s_N

- Assume observations are **independent**
 - Graph without links
- Ex: the prediction of rain for tomorrow is only based on the relative frequency of rainy days, while we ignore the influence of the previous day whether or not it rained.

MARKOV MODEL



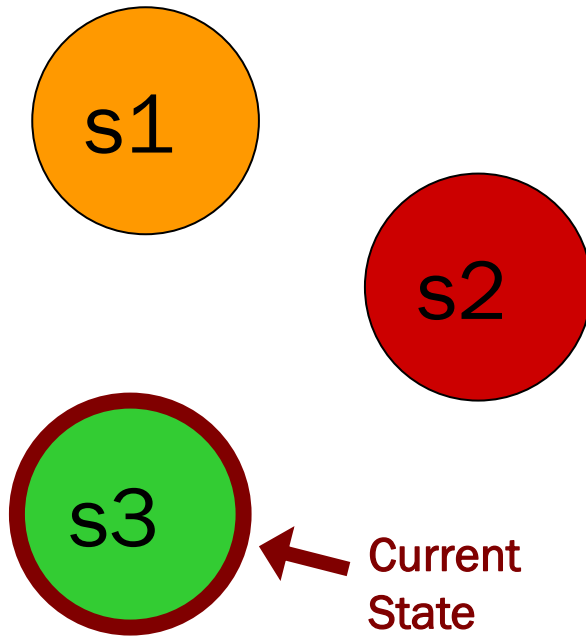
$N=3$

$t=1$

- Has N states , s_1, s_2, \dots, s_N
- It is characterized by discrete steps, $t=1, t=2, \dots$
- The probability of **starting from** a certain state is dictated by the **distribution**:
- $\{\pi_i\}$: $\pi_i = P(q_1 = s_i)$ with

$$1 \leq i \leq N, \quad \pi_i \geq 0 \quad \text{and} \quad \sum_{i=1}^N \pi_i = 1$$

MARKOV MODEL

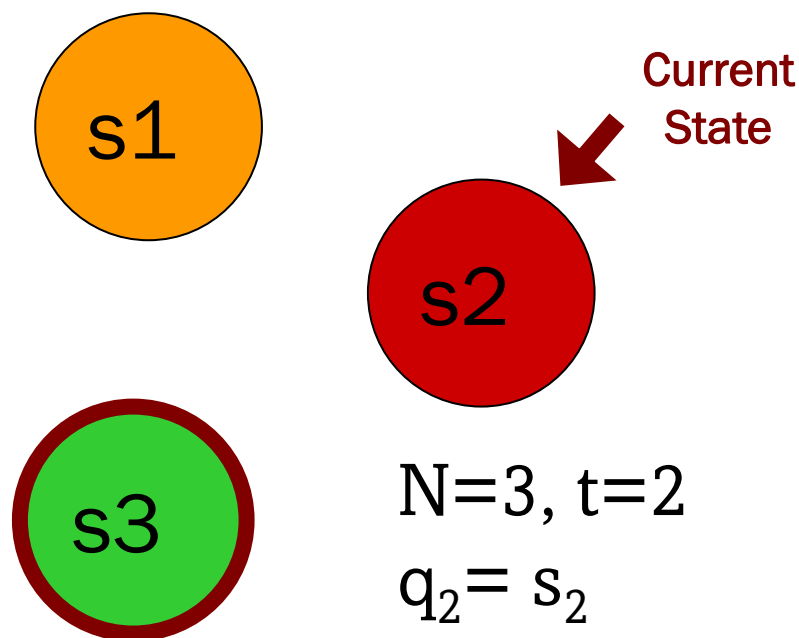


$N=3, t=1$

$q_1 = s_3$

- At the t -th instant the process is exactly in one of the available states, indicated by the variable q_t while $q_t \in \{s_1, s_2, \dots, s_N\}$
- At each iteration, the next state is chosen with a certain probability

MARKOV MODEL



$$\begin{aligned} P(q_{t+1}=s_1|q_t=s_1) &= 0 \\ P(q_{t+1}=s_2|q_t=s_1) &= 0 \\ P(q_{t+1}=s_3|q_t=s_1) &= 1 \end{aligned}$$

$$\begin{aligned} P(q_{t+1}=s_1|q_t=s_3) &= 1/3 \\ P(q_{t+1}=s_2|q_t=s_3) &= 2/3 \\ P(q_{t+1}=s_3|q_t=s_3) &= 0 \end{aligned}$$

$$\begin{aligned} P(q_{t+1}=s_1|q_t=s_2) &= 1/2 \\ P(q_{t+1}=s_2|q_t=s_2) &= 1/2 \\ P(q_{t+1}=s_3|q_t=s_2) &= 0 \end{aligned}$$

Current
State

- The probability is only determined by the **previous state** (*first-order markovianity*):

$$P(q_{t+1}=s_j | q_t=s_i, q_{t-1}=s_k, \dots, q_1=s_l) = P(q_{t+1}=s_j | q_t=s_i)$$

FIRST-ORDER MARKOV MODEL

- If the model is used to predict the next observation, the distribution of prediction will only depend on the **preceding observation** and **be independent of earlier observations**.

$$P(q_t = S^* | q_{t-1}, \dots, q_1) = P(q_t = S^* | q_{t-1})$$

- We can show the transition between states invariant over time as follows:

A=

$a_{1,1}$	$a_{1,2}$	$a_{1,3}$
$a_{2,1}$	$a_{2,2}$	$a_{2,3}$
$a_{3,1}$	$a_{3,1}$	$a_{3,3}$

$$a_{ij} = P(q_{t+1} = s_j \mid q_t = s_i)$$

MARKOV MODELS INGREDIENTS

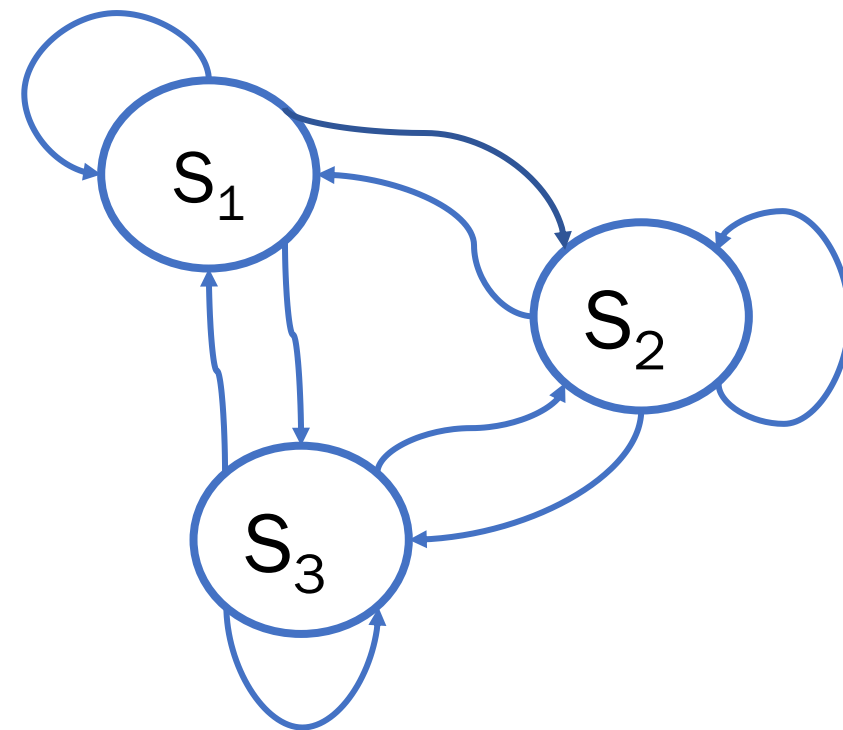
- A set of N **states** $\mathcal{S} = \{S_1, S_2, \dots, S_N\}$
- A sequence of states $\mathcal{Q} = \{q_1, q_2, \dots, q_T\}$

- A **transition probability matrix**

$$A = \{a_{ij} = P(q_t = S_j | q_{t-1} = S_i)\}$$

- An **initial probability distribution** over states

$$\Pi = \{\pi_i = P(q_1 = S_i)\}$$



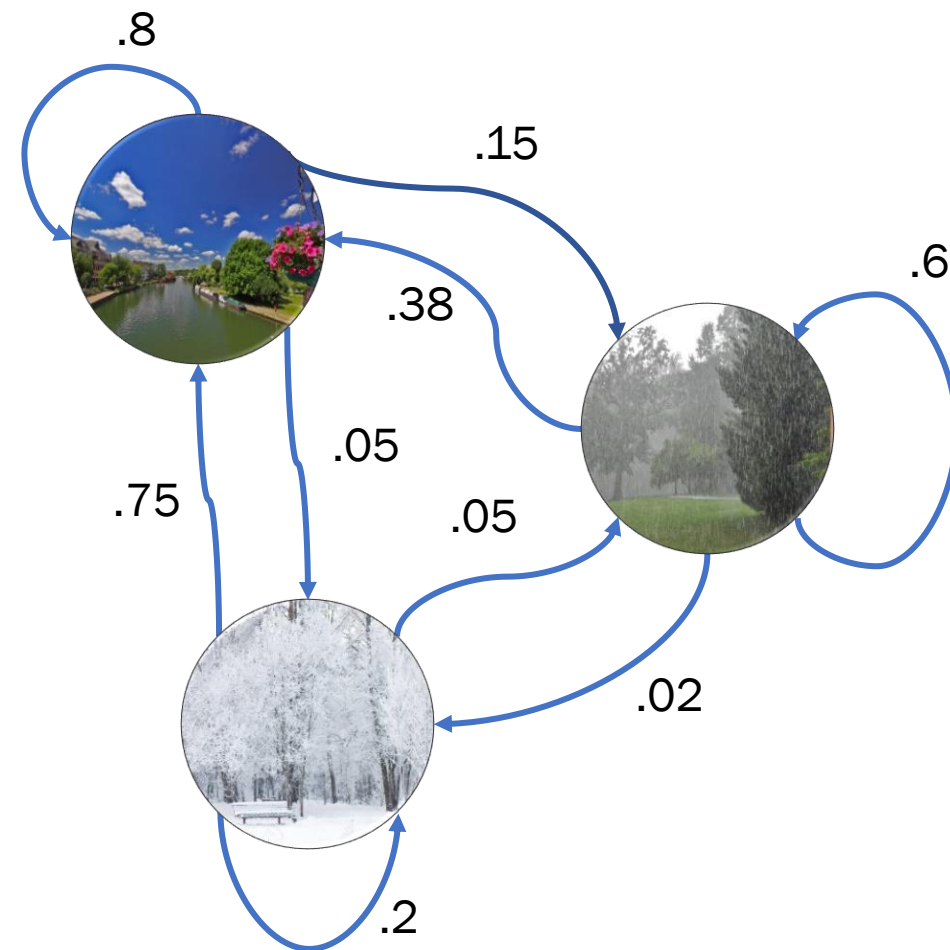
EXAMPLE

- **States:** $\{sunny, rainy, snowy\}$
- **Transition probability matrix:**

$$A = \begin{bmatrix} .8 & .15 & .05 \\ .38 & .6 & .02 \\ .75 & .05 & .2 \end{bmatrix}$$

- **Initial probability distribution:**

$$\Pi = [.7 \quad .25 \quad .05]$$



EXAMPLE

- **Compute the probability for the sequence:**



$$\begin{aligned} P &= P(Su) * P(R|Su) * P(R|R) * P(R|R) * P(Sn|R) * P(Sn|Sn) \\ &= 0.7 * 0.15 * 0.6 * 0.6 * 0.02 * 0.2 \end{aligned}$$

$$P = 0.0001512$$

MTH ORDER MARKOV MODEL

- **2nd order:** Each observation is influenced by the previous **two** observations
 - Conditional distribution of observation q_t depends on the values of **two previous observations** q_{t-1} and q_{t-2} .
- **Mth order Markov:** Conditional distribution for a particular variable depends on previous **M variables**.
 - More number of parameters.
 - First order: $P(q_t|q_{t-1})$ needs **S-1** parameters for each value of q_{t-1} for each of S states of q_t giving **K(K-1)** parameters
 - Mth order will need **K^{M-1}(K-1)** parameters
 - K represents the number of states that the variable q_t

LATENT VARIABLES

- Markov models are tractable (computationally manageable) but severely limited.
- We introduce **latent variables** to provide a more **general framework**.
- When **latent variables** are **discrete**: the model is called a ***Hidden Markov Model (HMM)***
 - *Each state is associated with a probability function that describes the probability that a certain output (**symbol**) is emitted by that state.*

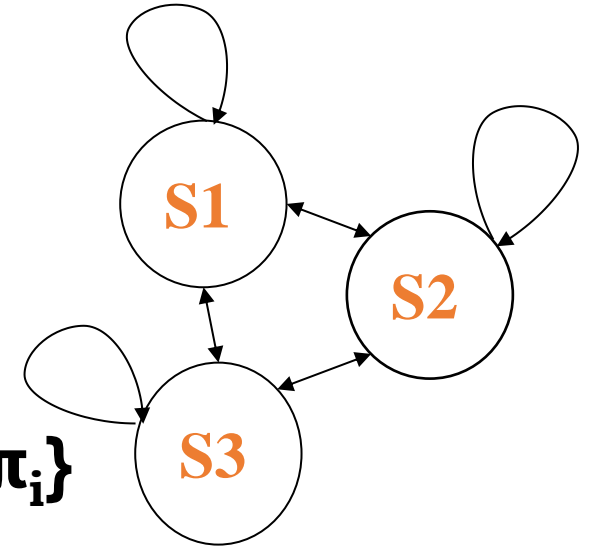
HMM: THE FORMAL DEFINITION

- An HMM consists of:
 - A set $S = \{s_1, s_2, \dots, s_N\}$ of **hidden states**
 - A **transition matrix** $A = \{a_{ij}\}$
between hidden states $1 \leq i, j \leq N$
 - An **initial distribution** over hidden states $\pi = \{\pi_i\}$

$$\pi = \begin{array}{|c|c|c|} \hline \pi_1 = 0.33 & \pi_2 = 0.33 & \pi_3 = 0.33 \\ \hline \end{array}$$

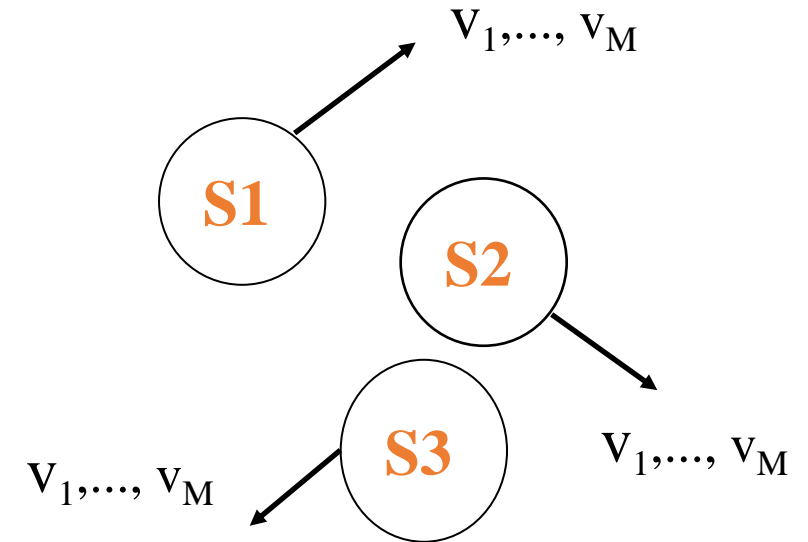
$$A =$$

$a_{11} = 0.1$	$a_{12} = 0.9$	$a_{13} = 0$
$a_{21} = 0.01$	$a_{22} = 0.2$	$a_{23} = 0.79$
$a_{31} = 1$	$a_{32} = 0$	$a_{33} = 0$



HMM: THE FORMAL DEFINITION

- A set $V = \{v_1, v_2, \dots, v_M\}$ of **observation symbols**
- A **probability distribution** on **observation symbols** $B = \{b_{jk}\}$, which *indicates the probability of emission of the symbol* v_k when the system state is s_j .



$B =$

$b_{11}=0.8$	$b_{21}=0.1$	$b_{31}=0.1$
$b_{12}= 0.1$	$b_{22}= 0.8$	$b_{32}= 0.1$
\vdots	\vdots	\vdots
$b_{1M}= 0.1$	$b_{2M}= 0.1$	$b_{3M}=0.8$

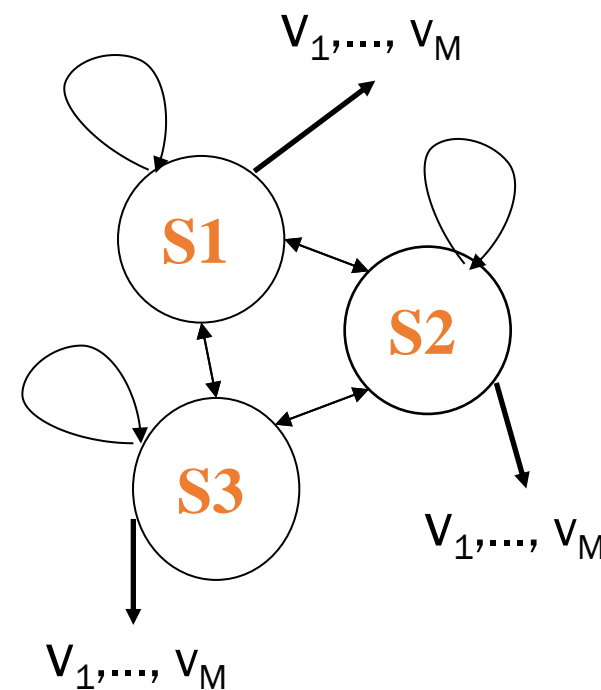
HMM: THE FORMAL DEFINITION

An **HMM** is denoted with a triple $\lambda = (A, \mathbf{B}, \pi)$

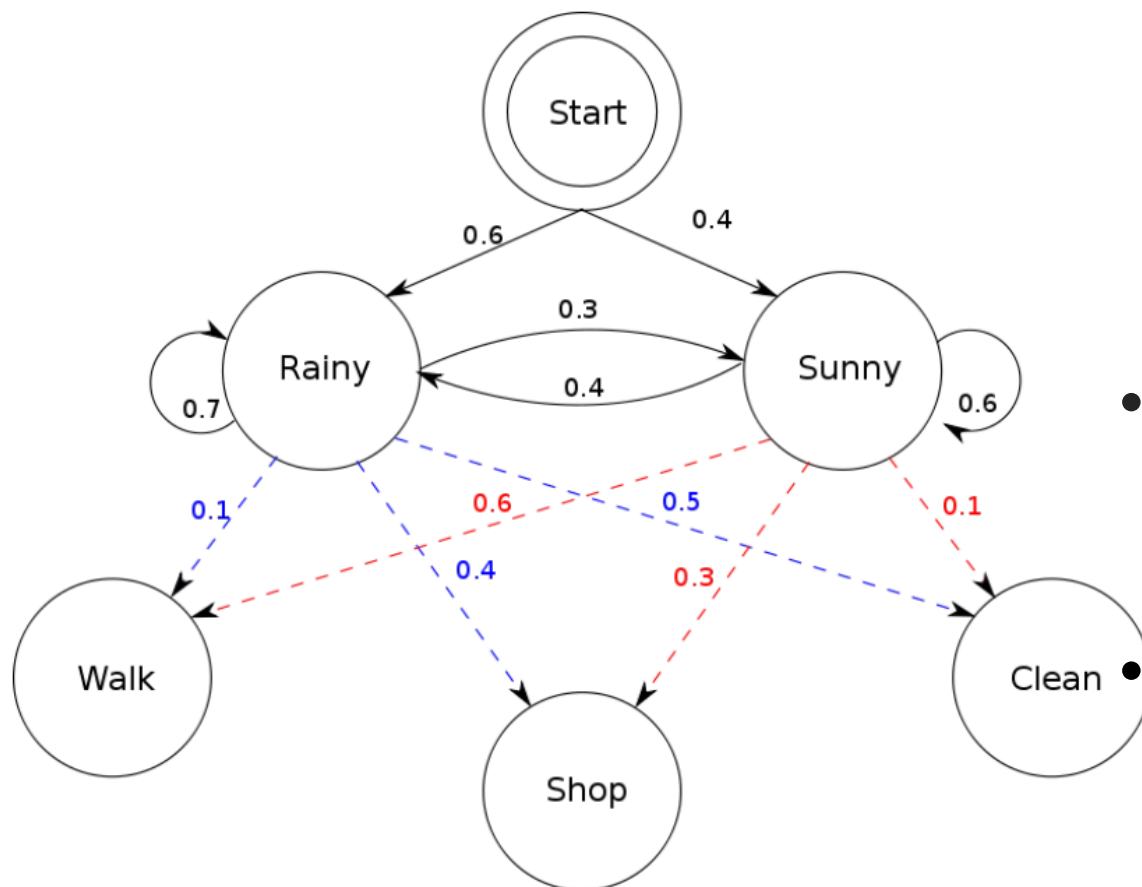
$$\pi = \begin{array}{|c|c|c|} \hline \pi_1 = 0.33 & \pi_2 = 0.33 & \pi_3 = 0.33 \\ \hline \end{array}$$

$$A = \begin{array}{|c|c|c|} \hline a_{11} = 0.1 & a_{12} = 0.9 & a_{13} = 0 \\ \hline a_{21} = 0.01 & a_{22} = 0.2 & a_{23} = 0.79 \\ \hline a_{31} = 1 & a_{32} = 0 & a_{33} = 0 \\ \hline \end{array}$$

$$B = \begin{array}{|c|c|c|} \hline b_{11} = 0.8 & b_{21} = 0.1 & b_{31} = 0.1 \\ \hline b_{12} = 0.1 & b_{22} = 0.8 & b_{32} = 0.1 \\ \hline \vdots & \vdots & \vdots \\ \hline b_{1M} = 0.1 & b_{2M} = 0.1 & b_{3M} = 0.8 \\ \hline \end{array}$$



HMM: EXAMPLE

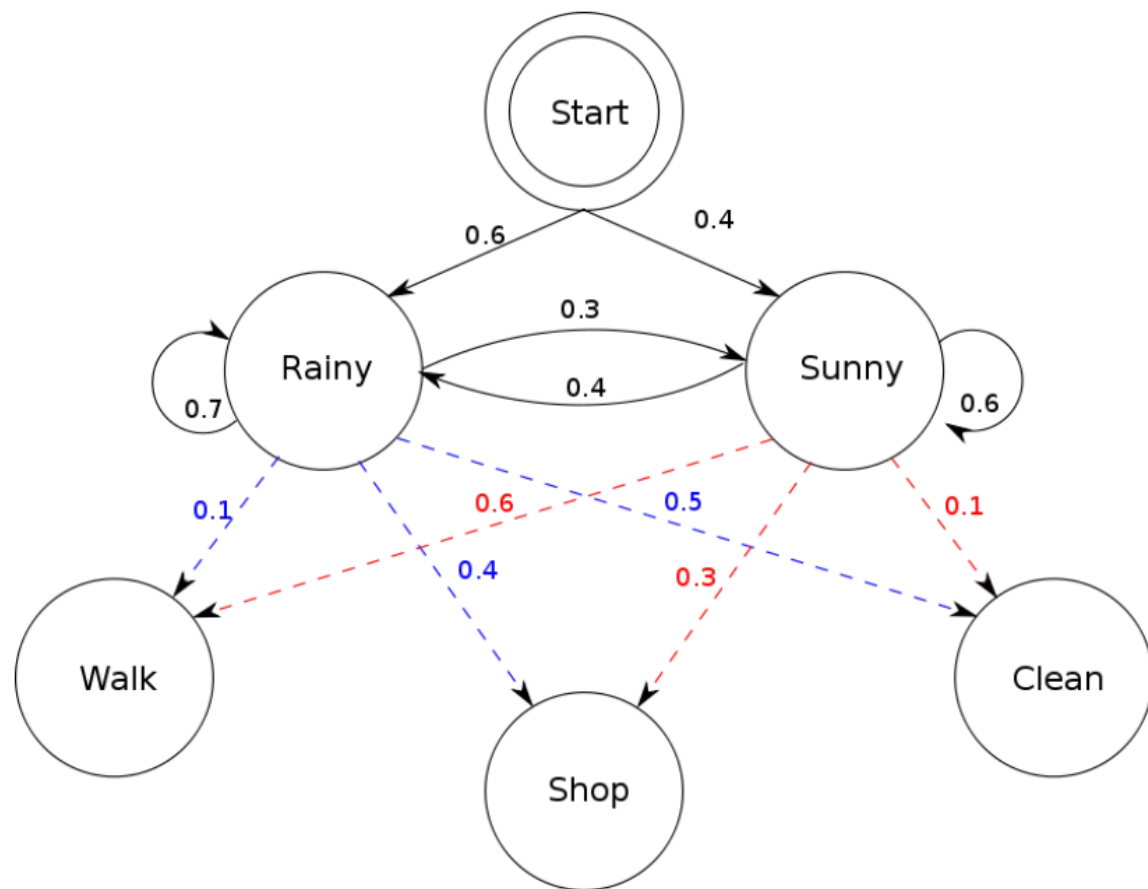


- Markov process is shown by the interaction between “Rainy” and “Sunny” in the below diagram and each of these are **HIDDEN STATES**.

- OBSERVATIONS** are known data and refers to “Walk”, “Shop”, and “Clean”.

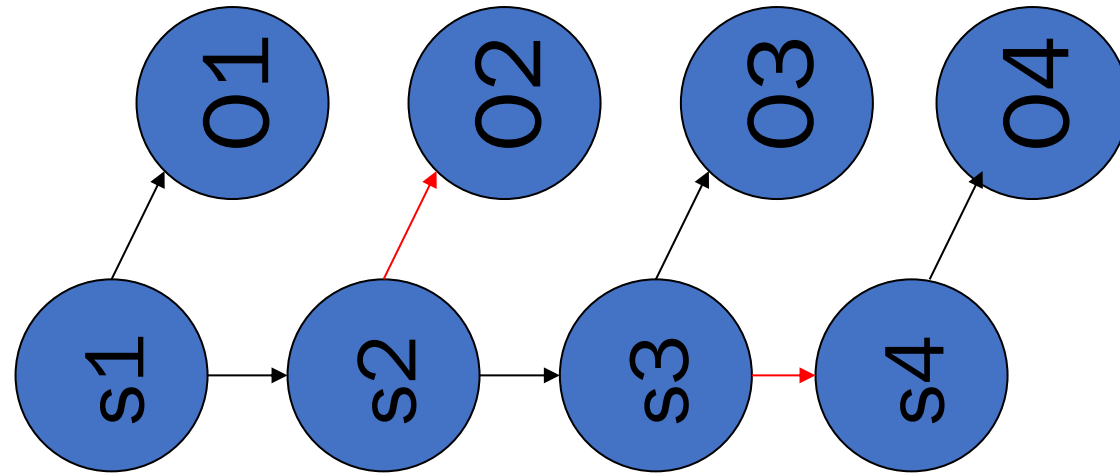
In ML sense, **observation** is our **training data**, and **the number of hidden states** is our **hyperparameter** for our model.

HMM: EXAMPLE



- **State transition probabilities** are the arrows pointing to each hidden state.
- **Observation probability matrix** are the **blue** and **red** arrows pointing to each observation from each hidden state.
 - The matrix B is row stochastic meaning the rows add up to 1.

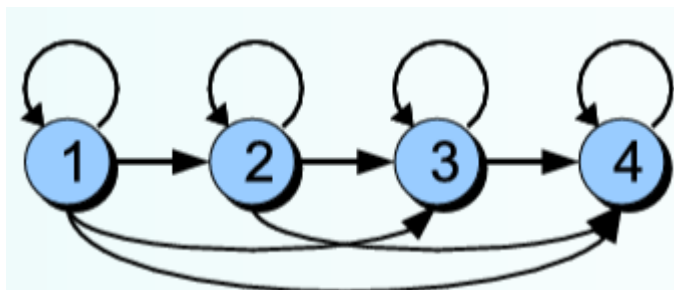
CONDITIONAL INDEPENDENCE WITH LATENT VARIABLES



Edges convey **conditional independence**

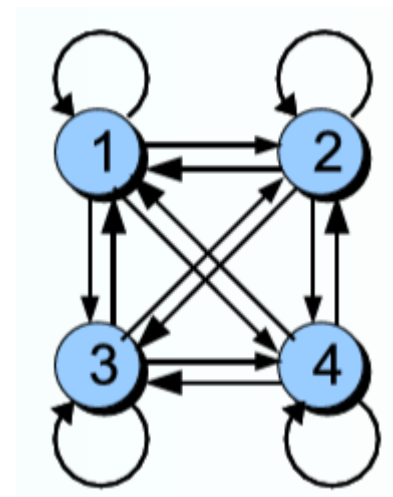
- O_2 is conditionally independent of everything else given s_2
- s_4 is conditionally independent of everything else given s_3

DIFFERENT TYPES OF HMM STRUCTURE



Left-to-right

- Imposing restrictions on transition matrix **A** such that $A_{jk}=0$ if $k < j$
- Once a state is vacated, it cannot be re-entered.



Fully connected

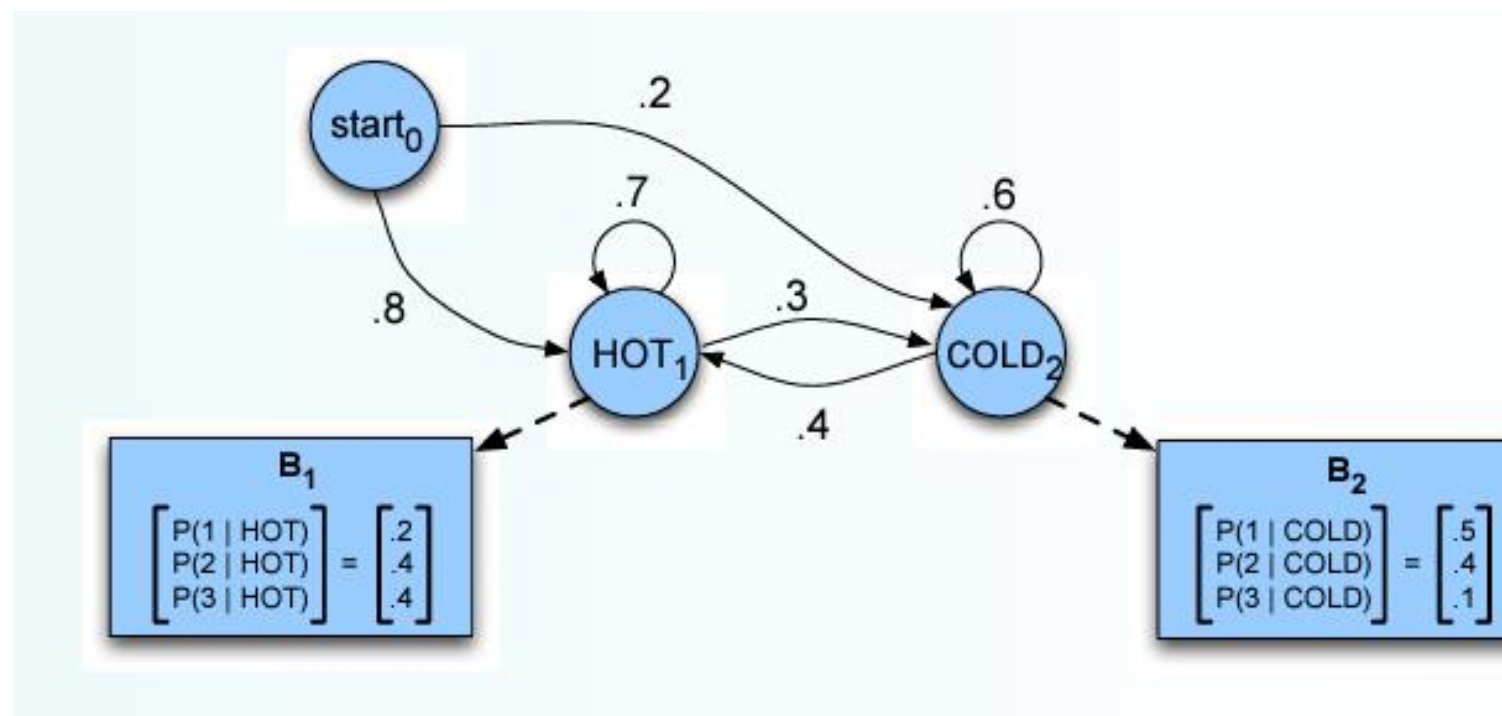


HMM USAGE: EXAMPLE

- Let's assume you are a climatologist in the year 2799
- You study global warming
- You do not have records of the weather in Verona for the summer of 2008
- But, you have the diary of Matteo Rossi who listed how many ice creams he ate every date that summer
- Your task is to figure out how hot it was in that summer.

HMM USAGE: EXAMPLE

- Given ice cream observation sequence (**B**): 1,2,3,2,2,2,3.....
- Produce: weather sequence (**A**): hot, cold, hot, hot, hot, cold.....
- HMM for ice cream:



KEY TASKS OF HMMS

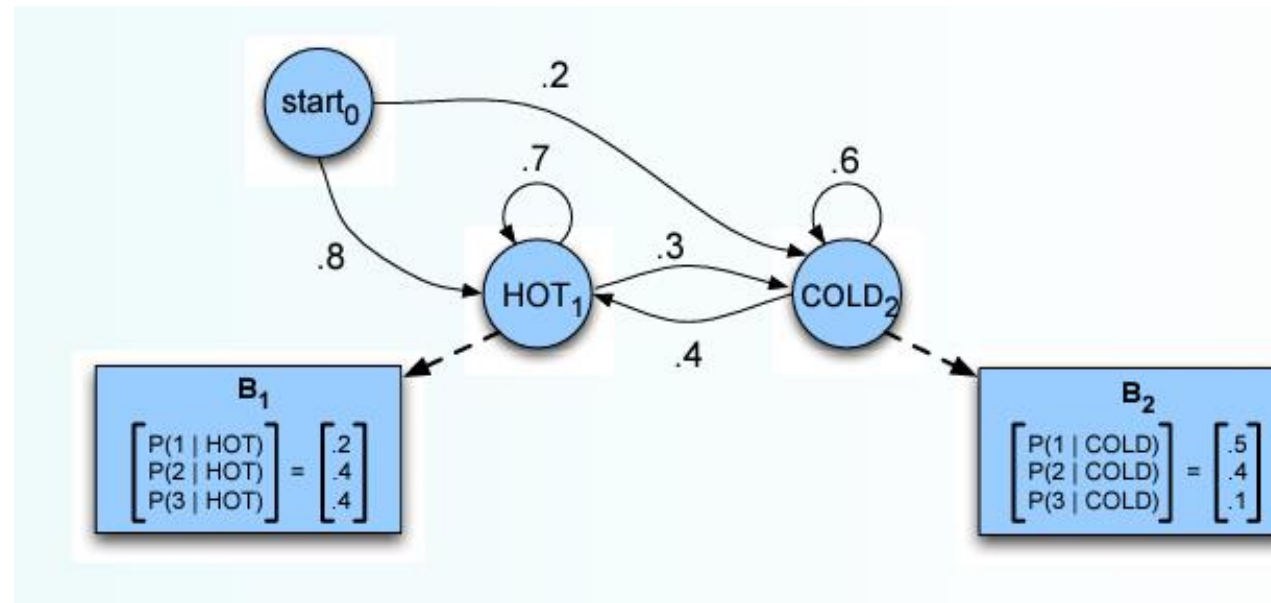
Task 1 (Evaluation or Likelihood): Given an HMM λ model and an observation sequence $\mathbf{O}=(O_1, O_2, \dots, O_t, \dots, O_T)$, computing $P(\mathbf{O} \mid \lambda)$, the probability of the observation sequence, given the model \rightarrow *forward procedure*

Task 2 (Decoding): Given an observation sequence \mathbf{O} and an HMM λ model, computing the most likely sequence of states $\mathbf{S}_1 \dots \mathbf{S}_T$ that generated $\mathbf{O} \rightarrow$ *Viterbi procedure*

Task 3 (Training): Given a set of observations $\{\mathbf{O}\}$, determining the best HMM model $\lambda=(\pi, A, B)$, i.e. the model for which $P(\mathbf{O} \mid \lambda)$ is maximized \rightarrow *Baum Welch procedure (forward-backward)*

TASK 1: COMPUTING THE OBSERVATION LIKELIHOOD

- Given an HMM λ model and an observation sequence $O=(O_1, O_2, \dots, O_t, \dots, O_T)$, determine the likelihood $P(O \mid \lambda)$.
- Given the following HMM:



- How likely is the sequence 3 1 3?



TASK 1: COMPUTING THE OBSERVATION LIKELIHOOD

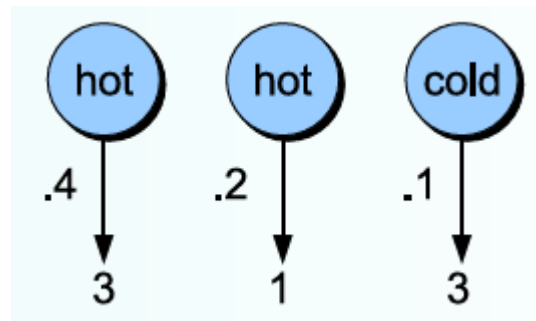
- To solve this, for Markov Chain, we **follow the states** and **multiply** the probabilities as we did before.
- But we do not know what the states are for an HMM!
- Thus, let's start with a simpler situation.
- (*A naïve approach*) Computing the **observation likelihood** for a **given** hidden state sequence
 - Assume we know the weather and want to predict how much ice cream Matteo would eat
 - E.g., $P(3 \ 1 \ 3 \mid H \ H \ C)$

TASK 1: COMPUTING THE OBSERVATION LIKELIHOOD

Computing likelihood of 3 1 3 given hidden state sequence:

$$P(O|Q) = \prod_{i=3}^T P(o_i|q_i)$$

$$P(3 \ 1 \ 3 \mid \text{hot hot cold}) = P(3 \mid \text{hot}) * P(1 \mid \text{hot}) * P(3 \mid \text{cold})$$

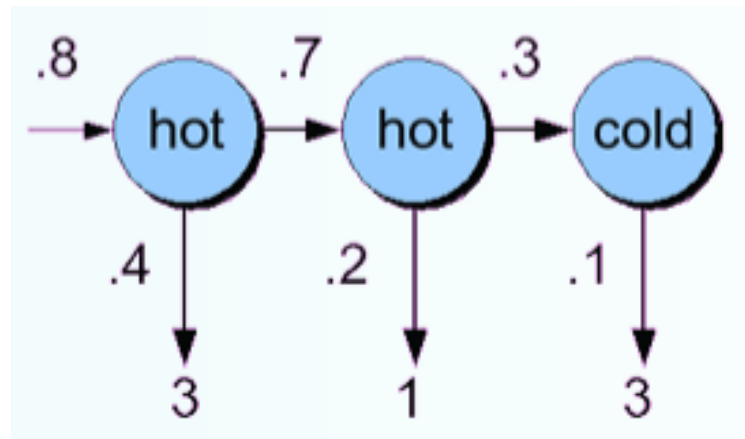


TASK 1: COMPUTING THE OBSERVATION LIKELIHOOD

Computing joint probability of observation and state sequence:

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

$$P(3 \ 1 \ 3, \text{hot hot cold}) = P(\text{hot} | \text{start}) * P(\text{hot} | \text{hot}) * P(\text{cold} | \text{hot}) * \\ P(3 | \text{hot}) * P(1 | \text{hot}) * P(3 | \text{cold})$$




TASK 1: COMPUTING THE OBSERVATION LIKELIHOOD

Computing total likelihood of 3 1 3:

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

We need to sum over

- **Hot hot cold** 
- Hot hot hot
- Hot cold hot
-



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



TASK 1: COMPUTING THE OBSERVATION LIKELIHOOD

hot hot hot

hot hot cold

hot cold hot

hot cold cold

cold hot hot

cold hot cold

cold cold hot

cold cold cold

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

- How many possible hidden state sequence are there for an HMM with N hidden states and sequence of T observation?
 - N^T
 - That is quite a big number, so we cannot just do separate computation for each hidden state sequence.
 - Instead, we can apply **forward algorithm**

TASK 1: THE FORWARD ALGORITHM

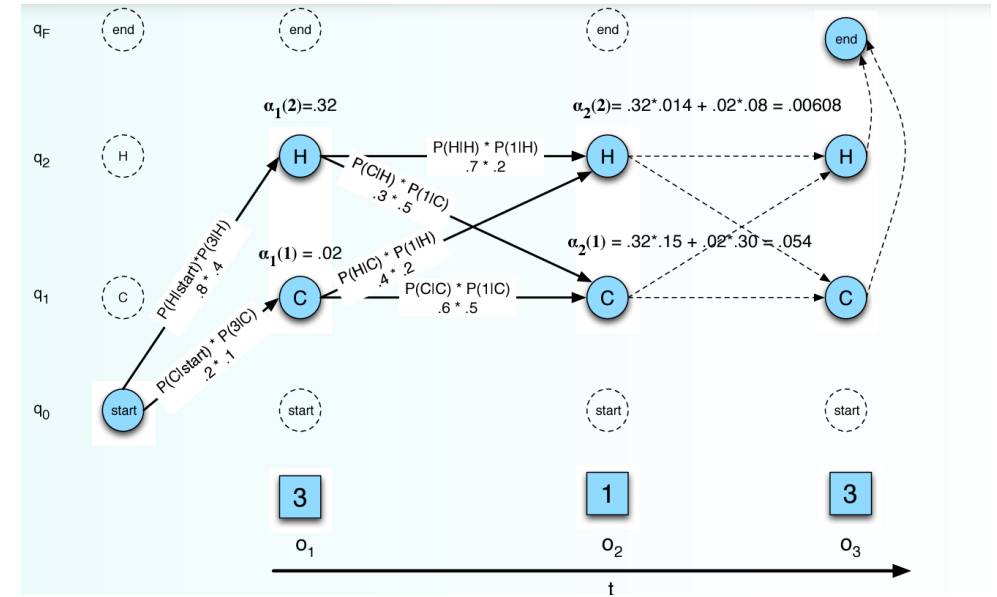
- Idea:
 - Compute the likelihood of the observation sequence
 - By summing over all possible hidden state sequences
 - But doing this efficiently
 - By folding all the sequences into a single **trellis**
- **Trellis:** a graph to visualize the computation of an optimal solution to a problem with overlapping subproblems. Helps to identify the optimal path or sequence of decision by considering all possible choices and their consequences.

TASK 1: THE FORWARD ALGORITHM

- The forward algorithm computes:

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

- Using **recursion**



- Each cell of the forward algorithm trellis $\alpha_t(j)$ represents the probability of being in **state j** , after seeing the first **t observations**
- Therefore, each cell expresses the following probability

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

TASK 1: THE FORWARD ALGORITHM

- **Initialization:**

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

- **Recursion** (note that the final state F is not emitting):

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

- **Termination:**

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

THE FORWARD TRELLIS

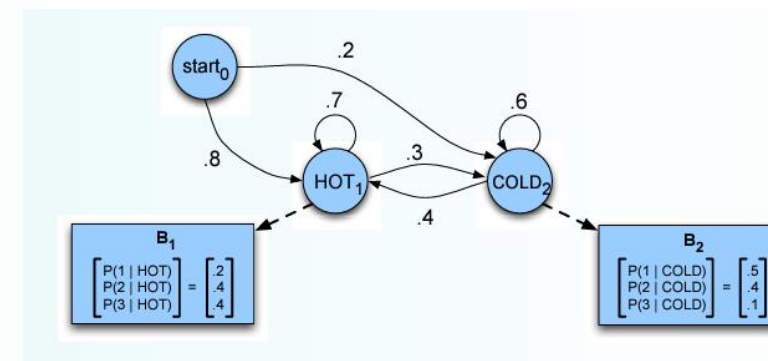
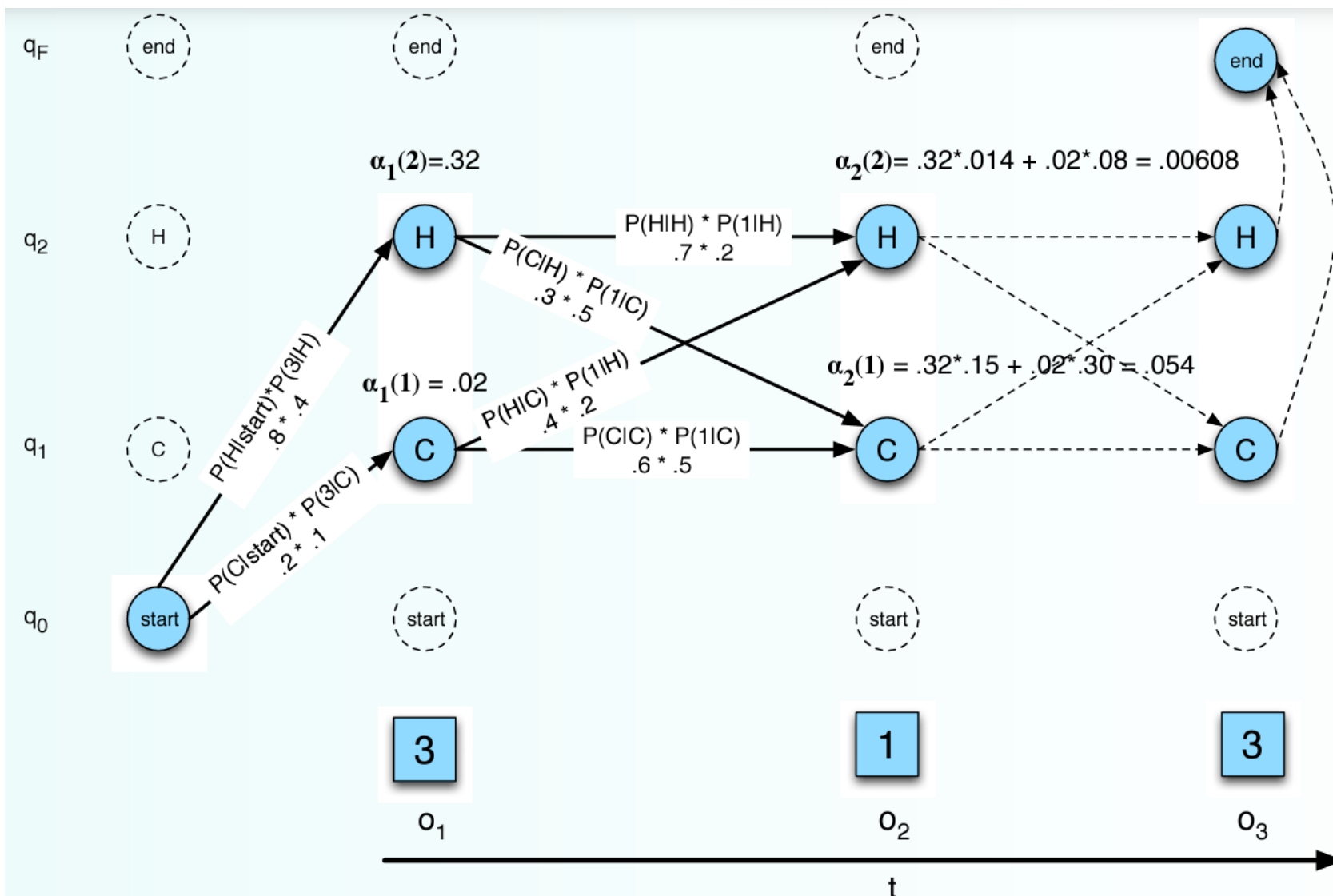


Fig: The computation
of $\alpha_t(j)$

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 \mathbf{o}_t given the current state j

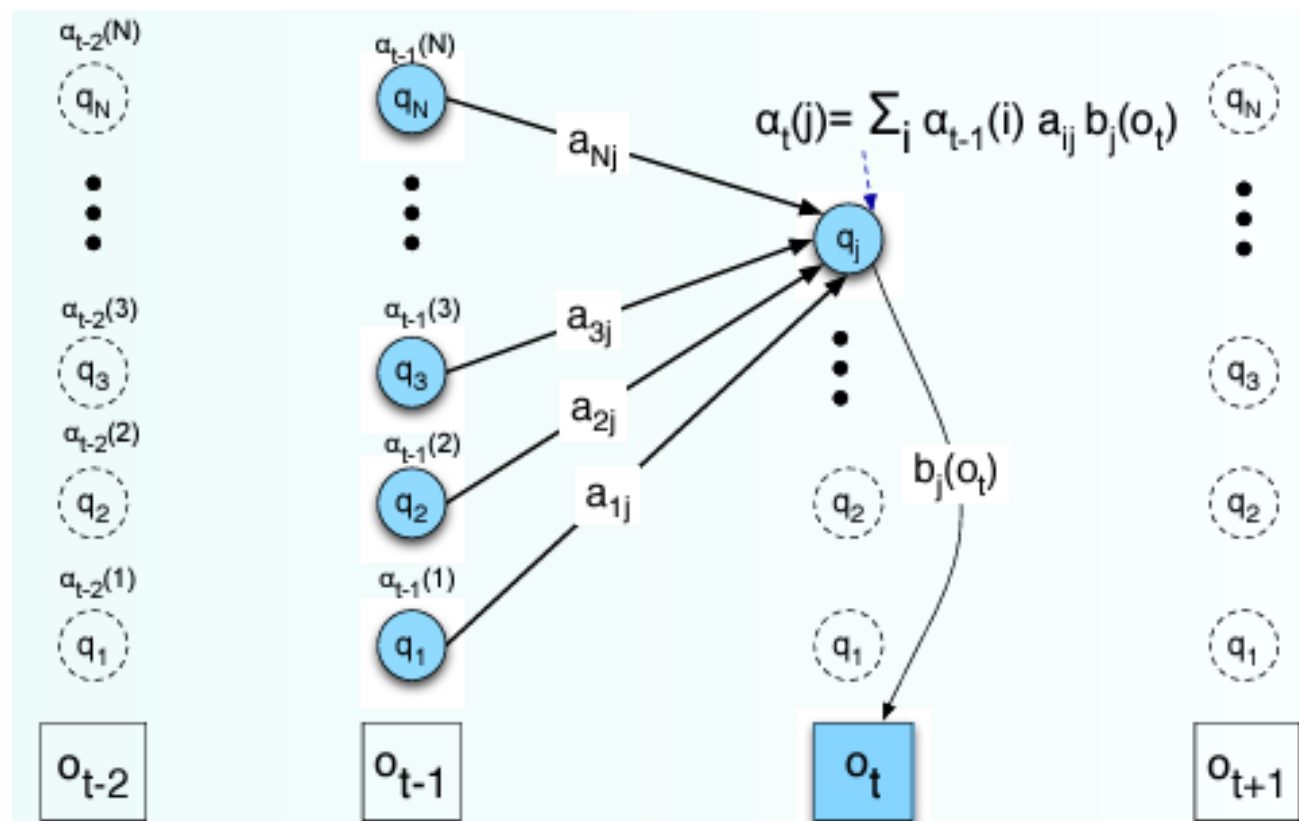


Fig: The computation of a single element $\alpha_t(j)$ in the trellis by summing all the previous values α_{t-1} , weighted by their transition probabilities a and multiplying by the observation probability $b_j(o_t)$

THE FORWARD PROBLEM 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**

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

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

return $forward[q_F, T]$

TASK 1: THE FORWARD ALGORITHM

- The forward algorithm efficiently computes the probability of observing the entire sequence O given the model λ in $\mathbf{O}(T \cdot N^2)$ time, where T is the sequence length and N is the number of states in the HMM.
- Consequently, we decreased the complexity from $\mathbf{N^T}$ to $\mathbf{T \cdot N^2}$.
 - The naïve solution recomputes the same subproblems (e.g., probabilities of partial sequences) repeatedly for different state sequences.
 - The Forward Algorithm stores and reuses these computations.

TASK 2: DECODING

- Task 1 is for observation sequence mainly.... But now we guess the **state sequences**.
- Given an observation sequence (e.g., 3 1 3), and an HMM $\lambda(A,B)$, the task of the decoder is to find the **best hidden** state sequence.
 - How to choose a state sequence $\mathbf{Q}=(q_1, q_2, q_3, q_4... q_T)$ that best explains the observations?

TASK 2: DECODING

- One possibility is to compute $P(O | Q)$ for each hidden state sequence Q
 - E.g., H H H, H H C, H C H
- Then, pick the highest one
- However, this has N^T complexity
- Instead
 - The **Viterbi algorithm** which is a dynamic programming algorithm like the forward algorithm can be used.
 - It uses a similar trellis to the forward algorithm.

VITERBI

- The aim is to compute the *joint probability of the observations sequence* together with *the best state sequence*:

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

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

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

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

VITERBI

- Note that the **Viterbi algorithm is identical to the forward algorithm** except that it takes the **max** over the previous path probabilities whereas the forward algorithm takes the **sum**.
- Importantly, Viterbi has an additional component, called **backpointers**.
 - The reason is that while the forward algorithm needs to produce an observation likelihood, the Viterbi algorithm must produce a probability and also the most likely state sequence.
 - We compute this best state sequence by keeping track of the path of hidden states that led to each state, and then at the end **backtracing** the best path to the beginning.

VITERBI

- Initialization:**

$$v_1(j) = a_{0,j} \cdot b_j(o_1)$$

$$bt_1(j) = 0 \quad (\text{no backtracking at time 1})$$

- Recursion** (once again final state (q_F) is not emitting):

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

- Termination:**

The best score: $P^* = v_T(q_F) = \max_{i=1}^N v_T(i) * a_{i,F}$

The start of backtrace: $q_T^* = bt_T(q_F) = \operatorname{argmax}_{i=1}^N v_T(i) * a_{i,F}$

VITERBI (RECAP)

backtracking for the observation sequence O^*

1. Start by determining the most likely final state at time T :

$$s_T^* = \arg \max_j v_T(j)$$

Assign the corresponding observation for this state:

$$o_T^* = o_{s_T^*}$$

2. For $t = T - 1, T - 2, \dots, 1$:

$$s_t^* = bt_{t+1}(s_{t+1}^*)$$

Assign the corresponding observation for this state:

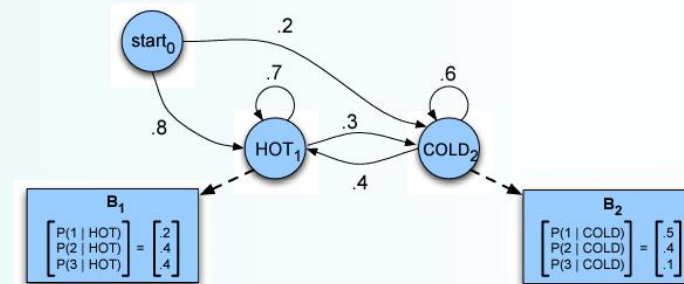
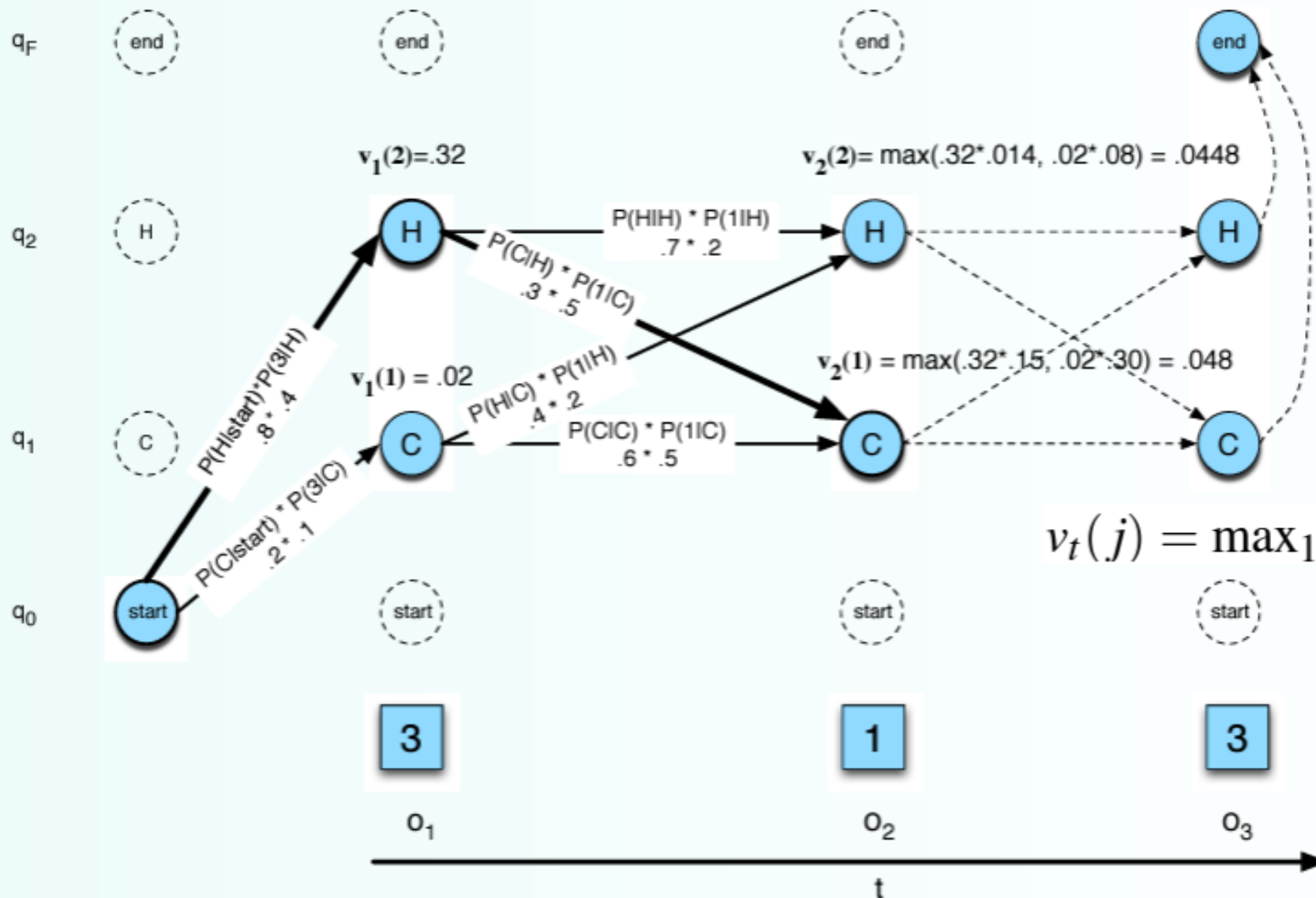
$$o_t^* = o_{s_t^*}$$

3. Collect $O^* = (o_1^*, o_2^*, \dots, o_T^*)$ as the most likely observation sequence.

The backtracking process ensures that the observation sequence reflects the most likely sequence of states according to the viterbi algorithm.

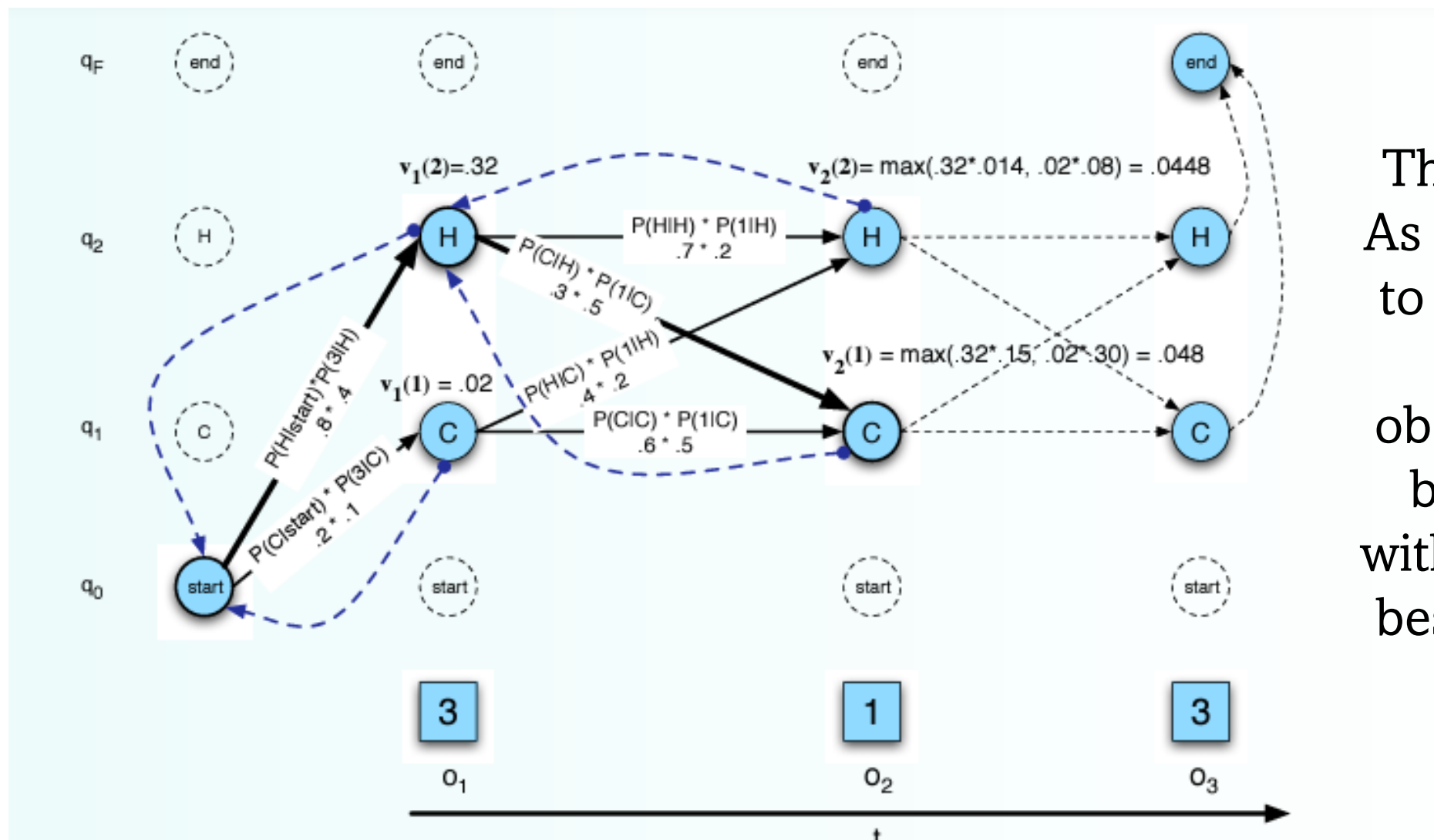
- While $v_T(j)$ provides the maximum probability for the final state, it alone does not store information about the preceding states that lead to this maximum probability.

VITERBI TRELLIS



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

VITERBI BACKTRACE



The Viterbi backtrace. As we extend each path to a new state account for the next observation, we keep a backpointer (shown with broken lines) to the best path that led us to this state.

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 $backpointer[q_F, T]$



TASK 3: HMM TRAINING

- Given an observation sequence O and the set of possible states in the HMM, learn the HMM parameters A and B .
- Input: An unlabelled sequence of observations \mathbf{O} and a vocabulary of potential hidden states \mathbf{Q} .
 - For the ice cream example, we would start with a sequence of observations $O = \{1, 3, 2, \dots\}$ and the set of hidden states $H(\text{hot})$ and $C(\text{cold})$.
- The standard algorithm for HMM training is the **forward-backward**, or **BaumWelch algorithm** (Baum, 1972),
 - a special case of the Expectation-Maximization (EM) algorithm (Dempster et al., 1977).

TASK 3: HMM TRAINING

- Let's start with a much simpler case of training a **fully visible Markov Model** in which we know both the temperature (hot or cold) and the ice cream count for every day:

3	3	2	1	1	2	1	2	3
Hot	hot	cold	cold	cold	cold	hot	hot	hot

- This would allow us to compute the HMM parameters by just maximum likelihood estimation from the training data.
- The initial states would be: $\pi_h = 1/3$ $\pi_c = 2/3$
- Next, we can directly compute **A matrix** by ignoring the final hidden states:
 - $P(\text{hot} | \text{hot}) = 2/3$ $P(\text{cold} | \text{hot}) = 1/3$
 - $P(\text{cold} | \text{cold}) = 2/3$ $P(\text{hot} | \text{cold}) = 1/3$

TASK 3: HMM TRAINING

- Let's start with a much simpler case of training a **fully visible Markov Model** in which we know both the temperature (hot or cold) and the ice cream count for every day:

3	3	2	1	1	2	1	2	3
Hot	hot	cold	cold	cold	cold	cold	hot	hot

- The B matrix:
 - $P(1 | \text{hot}) = 0/4 = 0$ $P(1 | \text{cold}) = 3/5 = .6$
 - $P(2 | \text{hot}) = 1/4 = .25$ $P(2 | \text{cold}) = 2/5 = .4$
 - $P(3 | \text{hot}) = 3/4 = .75$ $P(3 | \text{cold}) = 0$

TASK 3: HMM TRAINING

- To understand the Baum-Welch algorithm, we need to define **backward probability**, similar to what we defined as forward probability in Viterbi and Forward algorithms.
 - Computed in a similar manner to the forward algorithm.

$$\beta_t(i) = P(o_{t+1}, o_{t+2}, \dots, o_T \mid q_t = i, \lambda)$$

TASK 3: HMM TRAINING

1. Initialization:

$$\beta_T(i) = 1, \quad 1 \leq i \leq N$$

2. Recursion

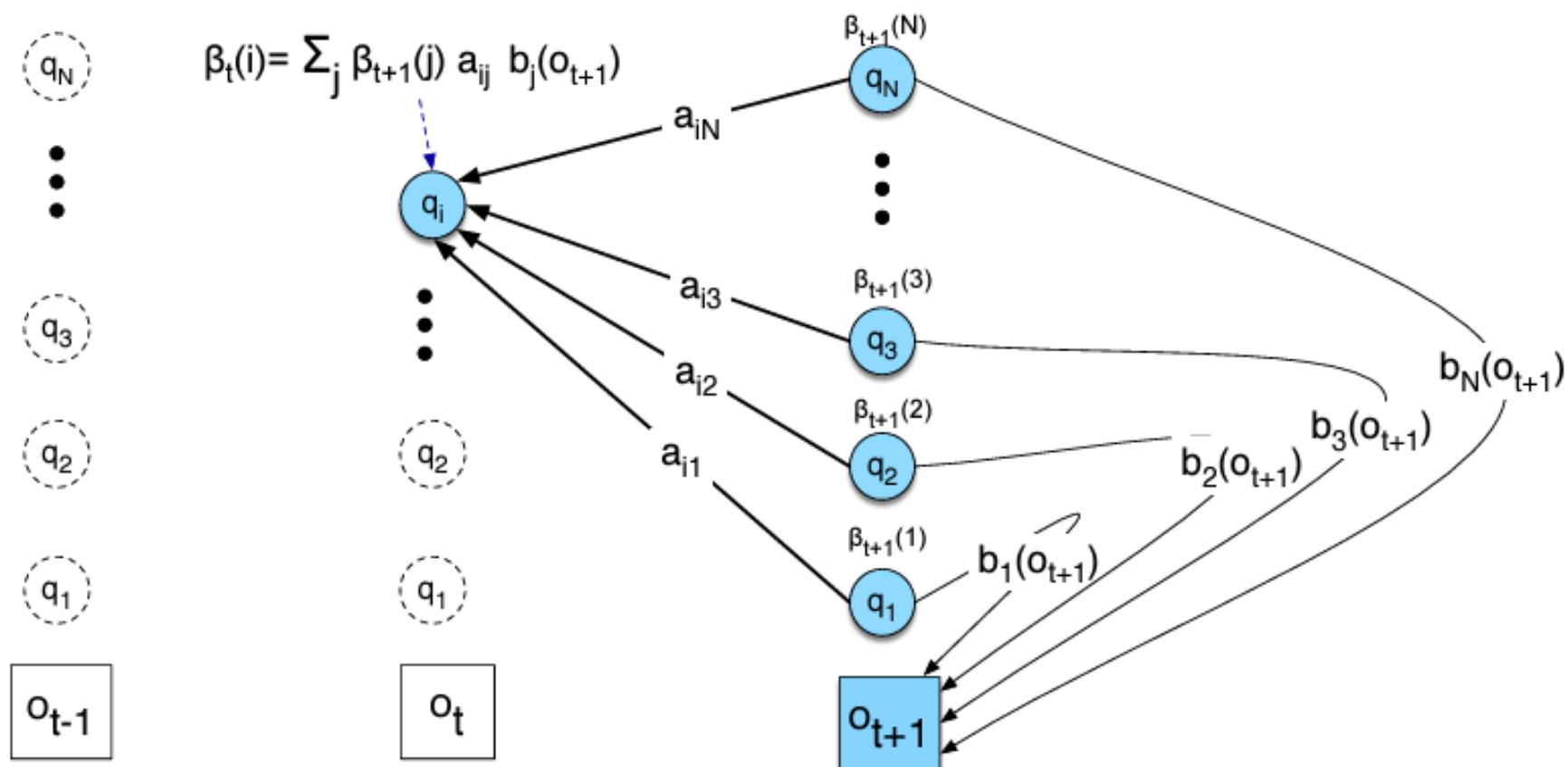
$$\beta_t(i) = \sum_{j=1}^N a_{ij} b_j(o_{t+1}) \beta_{t+1}(j), \quad 1 \leq i \leq N, 1 \leq t < T$$

3. Termination:

$$P(O|\lambda) = \sum_{j=1}^N \pi_j b_j(o_1) \beta_1(j)$$

The computation of $\beta_t(i)$ by summing all the successive values $\beta_{t+1}(j)$ weighted by their transition probabilities a_{ij} and their observation probabilities $b_j(o_{t+1})$.

TASK 3: HMM TRAINING



TASK 3: HMM TRAINING

- How the **forward** and **backward probabilities** can help to compute the transition probability a_{ij} and observation probability $b_i(o_t)$ from an observation sequence, even though the actual path taken through the model is hidden?
- Let's first see how to estimate \hat{a}_{ij} by a variant of simple maximum likelihood estimation:
$$\hat{a}_{ij} = \frac{\text{expected number of transitions from state } i \text{ to state } j}{\text{expected number of transitions from state } i}$$
- To compute the numerator, we need to use an intuition.

TASK 3: HMM TRAINING

- Let's first see how to estimate \hat{a}_{ij} by a variant of simple maximum likelihood estimation:

$$\hat{a}_{ij} = \frac{\text{expected number of transitions from state } i \text{ to state } j}{\text{expected number of transitions from state } i}$$

- To compute the numerator, we need to use an intuition.
 - Assume we had some estimate of the probability that a given transition $i \rightarrow j$ was taken at a particular point in time t in the observation sequence.
 - If we knew this probability for each particular time t , we could sum over all times t to estimate the total count for the transition $i \rightarrow j$.

TASK 3: HMM TRAINING

- Formally, we define the probability $\xi_t(ksi)$ as the probability of being in state i at time t and state j at time $t+1$, **given** the **entire** observation sequence and the HMM model:

$$\xi_t(i, j) = P(q_t = i, q_{t+1} = j \mid O, \lambda)$$

- To compute ξ_t , we first compute not-quiet- ξ_t which is the **joint probability** of being in state i at time t and state j at time $t+1$, along with observing the sequence O .

$$\xi_t(i, j) = \frac{\text{not-quiet-}\xi_t(i, j)}{P(O \mid \lambda)}$$

- Not-quiet- ξ_t avoids the intractable computation of $P(O \mid \lambda)$ directly by leveraging dynamic programming techniques.

TASK 3: HMM TRAINING

not-quite- $\xi_t(i, j) = P(q_t = i, q_{t+1} = j, O | \lambda)$

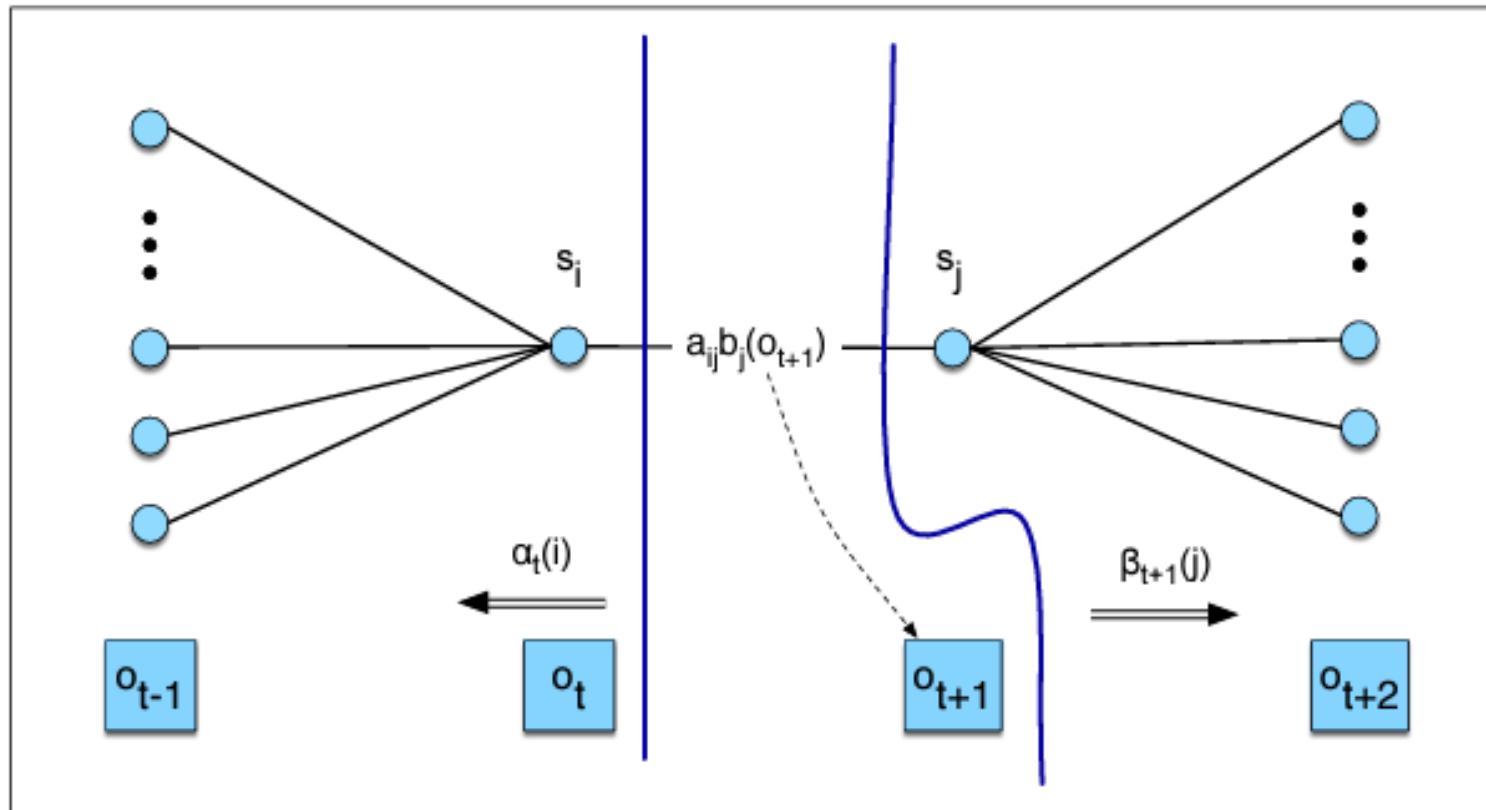


Fig. Computation of the joint probability of being in *state i* at *time t* and *state j* at *time t + 1*.

Various probabilities that need to be combined to produce $P(q_t = i, q_{t+1} = j, O | \lambda)$: the α and β probabilities, the **transition probability** a_{ij} , and the **observation probability** $b_j(o_{t+1})$.

TASK 3: HMM TRAINING

$$\text{not-quite-} \xi_t(i, j) = P(q_t = i, q_{t+1} = j, O | \lambda)$$

The various probabilities that go into computing **not-quite- ξ_t** : the transition probability for the arc in question, the **α probability** before the arc, the **β probability** after the arc, and the **observation probability** for the symbol just after the arc. These four are multiplied together to produce not-quite- ξ_t as follows:

$$\text{not-quite-} \xi_t(i, j) = \alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)$$

TASK 3: HMM TRAINING

- To compute ξ_t from **not-quite- ξ_t** , we follow the laws of probability and divide by $P(O | \lambda)$, since $P(X | Y, Z) = P(X, Y | Z) / P(Y | Z)$
- The probability of the observation given the model is simply the forward probability or backward probability of the whole utterance:

$$P(O | \lambda) = \sum_{i=1}^N \alpha_t(j) \beta_t(j)$$

- Thus,

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i, j)}{\sum_{t=1}^{T-1} \sum_{k=1}^N \xi_t(i, k)}$$

TASK 3: HMM TRAINING

- We also need a formula for recomputing the observation probability. This is the probability of a given symbol v_k from the observation vocabulary V , given a state j : $\hat{b}_j(v_k)$.
- We will do this by trying to compute

$$\hat{b}_j(v_k) = \frac{\text{expected number of times in state } j \text{ and observing symbol } v_k}{\text{expected number of times in state } j}$$

- For this, we need to know the probability of being in state j at time t , which is:

$$\gamma_t(j) = P(q_t = j | O, \lambda)$$

- That can be calculated by including the observation sequence in the probability:

$$\gamma_t(j) = \frac{P(q_t = j, O | \lambda)}{P(O | \lambda)}$$

TASK 3: HMM TRAINING

$$\gamma_t(j) = \frac{P(q_t = j, O|\lambda)}{P(O|\lambda)}$$

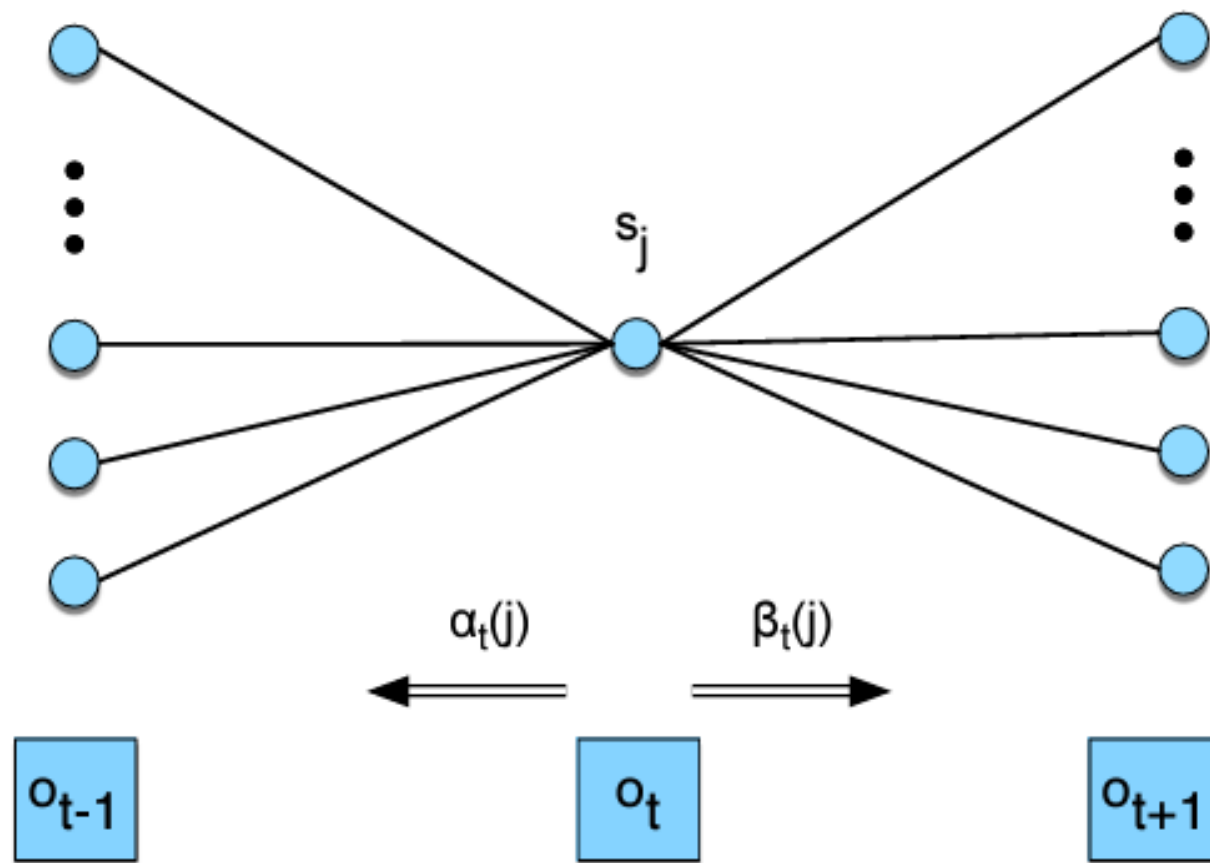


Fig. The computation of $\gamma_t(j)$, the probability of being in state j at time t . Note that γ is really a degenerate case of ξ .

$$\gamma_t(j) = \frac{\alpha_t(j)\beta_t(j)}{P(O|\lambda)}$$

The numerator is the product of the forward and backward probability.

TASK 3: HMM TRAINING

- Now, we can compute b. For the numerator, we sum $\gamma_t(j)$ for all time steps t in which the observation o_t is the symbol v_k that we are interested in.
- For the denominator, we sum $\gamma_t(j)$ over all time steps t . The result is the percentage of the times that we were in state j and saw symbol v_k :

$$\hat{b}_j(v_k) = \frac{\sum_{t=1}^T \text{s.t. } O_t = v_k \gamma_t(j)}{\sum_{t=1}^T \gamma_t(j)}$$

sum over all t for which
the observation at time t
was v_k

TASK 3: HMM TRAINING

- By assuming that we already have a previous estimate of A and B, we now have re-estimates of A and B from observation sequence O :

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i, j)}{\sum_{t=1}^{T-1} \sum_{k=1}^N \xi_t(i, k)} \quad \hat{b}_j(v_k) = \frac{\sum_{t=1}^T \mathbb{1}_{\{O_t=v_k\}} \gamma_t(j)}{\sum_{t=1}^T \gamma_t(j)}$$

- These two re-estimations form the core of the iterative forward-backward algorithm, which starts with some initial estimate of the HMM parameters. We then iteratively run two steps:
 - E-step**
 - M-step**



TASK 3: HMM TRAINING

- In the **E-step**, we compute the expected state occupancy count γ and the expected state transition count ξ from the earlier A and B probabilities.
- In the **M-step**, we use γ and ξ to recompute new A and B probabilities.

TASK 3: HMM TRAINING

function FORWARD-BACKWARD(*observations* of len T , *output vocabulary* V , *hidden state set* Q) **returns** $HMM=(A,B)$

initialize A and B

iterate until convergence

E-step

$$\gamma_t(j) = \frac{\alpha_t(j)\beta_t(j)}{\alpha_T(q_F)} \quad \forall t \text{ and } j$$

$$\xi_t(i, j) = \frac{\alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)}{\alpha_T(q_F)} \quad \forall t, i, \text{ and } j$$

M-step

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i, j)}{\sum_{t=1}^{T-1} \sum_{k=1}^N \xi_t(i, k)}$$

$$\hat{b}_j(v_k) = \frac{\sum_{t=1 \text{ s.t. } O_t=v_k}^T \gamma_t(j)}{\sum_{t=1}^T \gamma_t(j)}$$

return A, B

- Although in principle the forward-backward algorithm can do completely **unsupervised learning** of the A and B parameters, in practice the initial conditions are very important. For this reason the algorithm is often given extra information.
- For example, for HMM-based speech recognition, the HMM structure is often set by hand, and only the emission (B) and (non-zero) A transition probabilities are trained from a set of observation sequences O .

SUMMARY

- HMMs are a way of relating a sequence of observations to a sequence of hidden classes or hidden states that explain the observations.
- The process of discovering the sequence of hidden states, given the sequence of observations, is known as **decoding** or **inference**. The Viterbi algorithm is commonly used for decoding.
- The parameters of an HMM are the A transition probability matrix and the B observation likelihood matrix.
 - Both can be trained with the Baum-Welch or forward-backward algorithm.



That's all

Cigdem Beyan
cigdem.beyan@univr.it
<https://cbeyan.github.io/>