

Faculty for Informatics

Technical  
University  
of Munich



# Natural Language Processing IN2361

PD Dr. Georg Groh

Social Computing  
Research Group

# Chapter A

## Hidden Markov Models

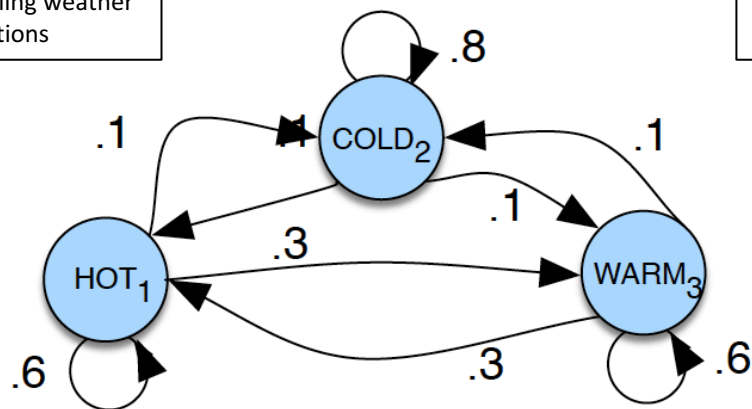
- content is based on [1]
- certain elements (e.g. equations or tables) were taken over or taken over in a modified form from [1]
- citations of [1] or from [1] are omitted for legibility
- errors are fully in the responsibility of Georg Groh
- BIG thanks to Dan and James for a great book!

# Hidden Markov Models

- HMM: probabilistic **sequence classifier**: input: sequence of observations; output: probability distribution over sequence of labels
- still used a lot in speech recognition and text-based NLP
- often “traditional” ML methods still state of the art, especially when data is scarce (if you do not have enough gun-powder, trying to shoot at stuff with a big cannon (complicated NNs) will just produce a mere “poop”

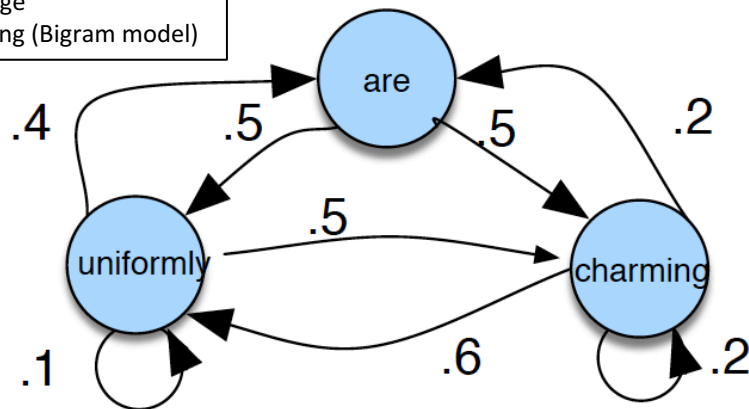
# (First Order) Markov Chains

Modeling weather transitions



(a)

Language Modeling (Bigram model)



(b)

(first order)

**Markov Assumption:**  $P(q_i = a | q_1 \dots q_{i-1}) = P(q_i = a | q_{i-1})$

$Q = q_1 q_2 \dots q_N$

a set of  $N$  states

$A = a_{11} a_{12} \dots a_{n1} \dots a_{nn}$

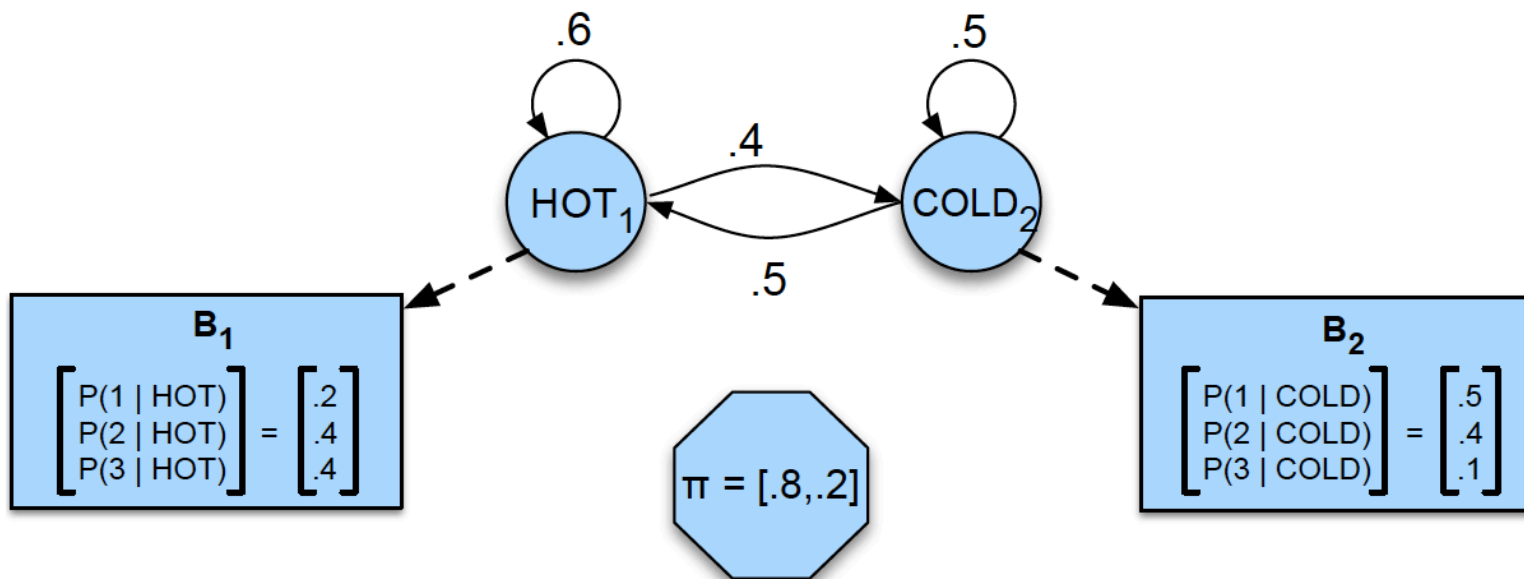
a **transition probability matrix**  $A$ , each  $a_{ij}$  representing the probability of moving from state  $i$  to state  $j$ , s.t.  $\sum_{j=1}^n a_{ij} = 1 \quad \forall i$

$\pi = \pi_1, \pi_2, \dots, \pi_N$

an **initial probability distribution** over states.  $\pi_i$  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,  $\sum_{i=1}^n \pi_i = 1$

# Hidden Markov Models

- If we **know transition matrix**: it's **easy** to compute probability of a given observed sequence:  $P(q_t, q_{t+1}, \dots, q_{t+m}) = P(q_t) a_{q_t q_{t+1}} a_{q_{t+1} q_{t+2}}, \dots, a_{q_{t+m-1} q_{t+m}}$
- often: **states** are **not observed** (“hidden”, “causal”, latent variables).  
example: part of speech (POS) tags  $\rightarrow$  **HMM**
- HMM example: **observe**: # of ice creams eaten; **hidden** states: cold weather, hot weather:



# Hidden Markov Models

**Markov Assumption:**  $P(q_i|q_1 \dots q_{i-1}) = P(q_i|q_{i-1})$

**Output Independence:**  $P(o_i|q_1 \dots q_i, \dots, q_T, o_1, \dots, o_i, \dots, o_T) = P(o_i|q_i)$

$Q = \{q_1 q_2 \dots q_N\}$  a set of  $N$  **states**

$A = a_{11} \dots a_{ij} \dots a_{NN}$  a **transition probability matrix**  $A$ , each  $a_{ij}$  representing the probability of moving from state  $i$  to state  $j$ , s.t.  $\sum_{j=1}^N a_{ij} = 1 \quad \forall i$

$$a_{ij} = P(q_t = j | q_{t-1} = i)$$

$O = (o_1 o_2 \dots o_T) = o_{1:T}$  a sequence of  $T$  **observations**, each one drawn from a vocabulary  $V = v_1, v_2, \dots, v_V$

$B = b_i(o_t)$  a sequence of **observation likelihoods**, also called **emission probabilities**, each expressing the probability of an observation  $o_t$  being generated from a state  $i$

$$b_i(o_t) = P(o_t | q_t = i)$$

# Hidden Markov Models

Markov Assumption:  $P(q_i | q_1 \dots q_{i-1}) = P(q_i | q_{i-1})$

Output Independence:  $P(o_i | a_1 \dots a_i \dots a_T, o_1 \dots o_{i-1} \dots o_{i+1} \dots o_T) = P(o_i | a_i)$

$Q = \{$

$A = \{$

$O = \{$

$B = b_i(o_t)$

**remark:**

the **wording** in Jurafsky and the literature on HMMs is sometimes a bit **blurry** in distinguishing whether something is a likelihood and whether something is a probability.

correct:  $P(x|\theta)$  is a “**probability of  $x$**  given  $\theta$ ” and a “**likelihood of  $\theta$** ” (given  $x$ ) so here  $b_i(o_t) = P(o_t | q_t = i)$  should be called a “likelihood of the state” or a “probability of the observation” given the state.

a sequence of **observation likelihoods**, also called **emission probabilities**, each expressing the probability of an observation  $o_t$  being generated from a state  $i$

$$b_i(o_t) = P(o_t | q_t = i)$$

# Fundamental HMM Problems

- **Problem 1 (Likelihood):** Given an HMM  $\lambda = (A, B)$  and an observation sequence  $O$ , determine the likelihood  $P(O|\lambda)$ .
  - other formulation: compute  $P(o_{1:T}|\lambda)$
  - solution: Forward algorithm
- **Problem 2 (Decoding):** Given an observation sequence  $O$  and an HMM  $\lambda = (A, B)$ , discover the best hidden state sequence  $Q$ .
  - other formulation: compute  $\operatorname{argmax}_{q_{1:T}} P(q_{1:T}|o_{1:T}, \lambda)$
  - solution: Viterbi algorithm
- **Problem 3 (Learning):** Given an observation sequence  $O$  and the set of states in the HMM, learn the HMM parameters  $A$  and  $B$ .
  - other formulation: compute  $\operatorname{argmax}_{A, B} P(o_{1:T}|\lambda(A, B, Q))$
  - solution: Forward-Backward (Baum Welch) algorithm (an instance of Expectation Maximization (EM))
- **Other problems:**
  - filtering / prediction:  $P(q_{T+k}|o_{1:T}, \lambda)$  (solution: forward algorithm variant);
  - smoothing:  $P(q_{T-k}|o_{1:T}, \lambda)$  (solution: backward algorithm)



# Fundamental HMM Problems

- **Problem 1 (Likelihood):** Given an HMM  $\lambda = (A, B)$  and an observation sequence  $O$ , determine the likelihood  $P(O|\lambda)$ .

**remark:**

the **wording** in Jurafsky and the literature on HMMs is sometimes a bit **blurry** in distinguishing whether something is a likelihood and whether something is a probability.

correct:  $P(x|\theta)$  is a “**probability of  $x$**  given  $\theta$ ” and a “**likelihood of  $\theta$** ” (given  $x$ )  
so here  $b_i(o_t) = P(o_t|q_t = i)$  should be called a “likelihood of the state” or a “probability of the observation” given the state.

on: compute  $P(o_{1:T}|\lambda)$   
d algorithm

Give  
( $A, B$

n: co  
algor

**remark:**

do not mix up  $Q = \{q_1, q_2, \dots, q_N\}$  (the set of states of/in the HMM, (which may better have been denoted as  $Q = \{q^{(1)}, q^{(2)}, \dots, q^{(N)}\}$ )) which is meant below, and  $Q = q_{1:t} = (q_1, q_2, \dots, q_t)$  (a sequence of states corresponding to a sequence of observations  $O = o_{1:t} = (o_1, o_2, \dots, o_t)$  at times  $t = 1, t = 2, \dots, t = t$ )

- **Problem 3 (Learning):** Given an observation sequence  $O$  and the set of states in the HMM, learn the HMM parameters  $A$  and  $B$ .

- other formulation: compute  $\operatorname{argmax}_{A,B} P(o_{1:T}|\lambda(A, B, Q))$
- solution: Forward-Backward / Baum Welch algorithm  
(an instance of Expectation Maximization (EM))

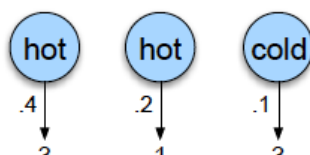
- **Other problems:**

- filtering / prediction:  $P(q_{T+k}|o_{1:T}, \lambda)$  (solution: forward algorithm variant);
- smoothing:  $P(q_{T-k}|o_{1:T}, \lambda)$  (solution: backward algorithm)

# The Forward Algorithm

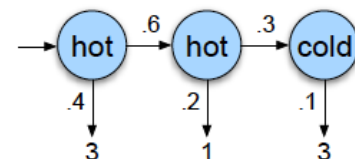
- if we knew  $O = o_{1:T}$  AND  $Q = q_{1:T}$ , we could simply compute

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

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


- we do however not know  $Q = q_{1:T} \rightarrow$  use the **joint**

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



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

and **sum out**  $Q$ :

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

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

# The Forward Algorithm

- Caveat: for a set of states with  $N$  states, there are  $N^T$  such combinations of state sequences in this example  $\rightarrow$  previous suggestion is **inefficient**
- $\rightarrow$  use **dynamic programming** (store intermediate results) in recursively formulated **Forward** algorithm  $\rightarrow$  takes only  $O(N^2 T)$  time

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

1. Initialization:

$$\alpha_1(j) = \pi_j b_j(o_1) \quad 1 \leq j \leq N$$

2. Recursion:

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

3. Termination:

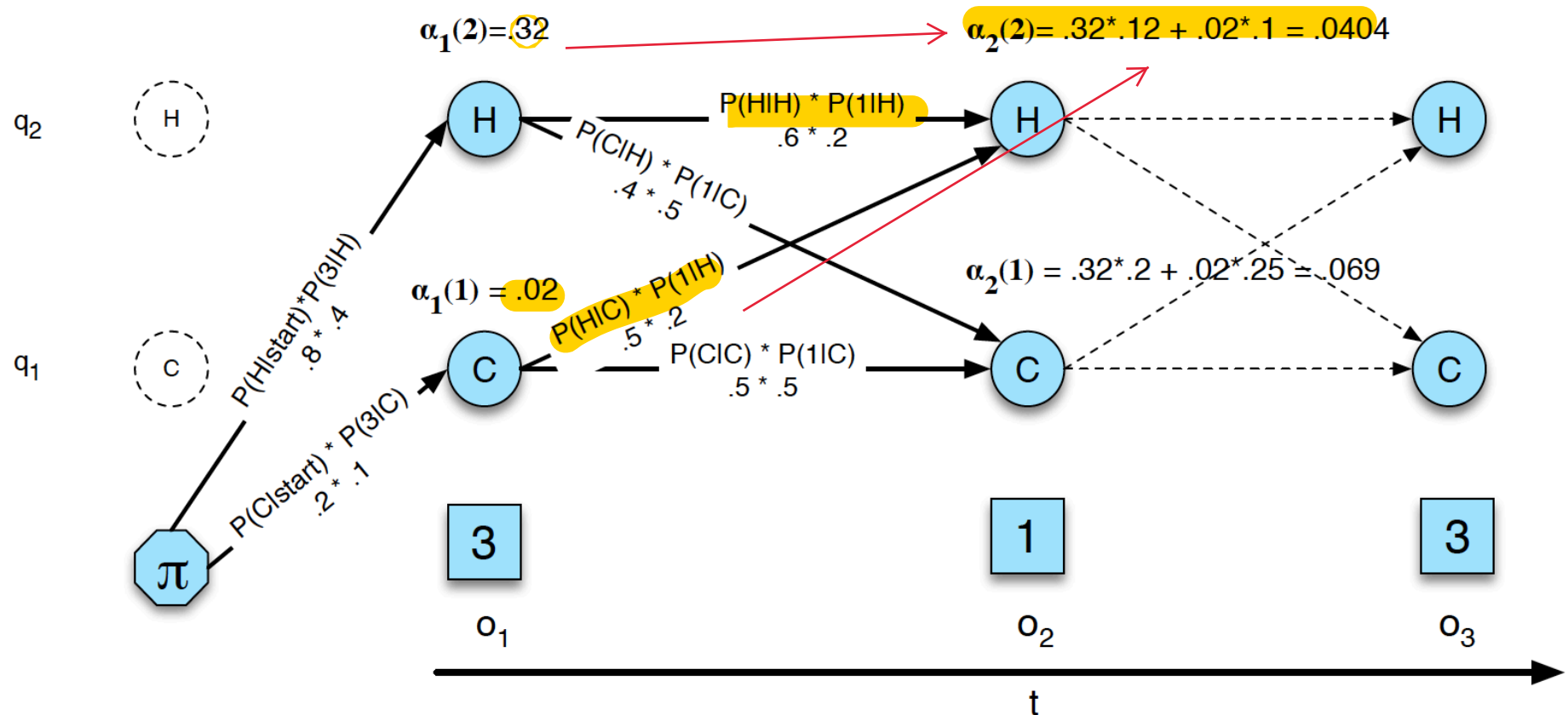
$$P(O | \lambda) = \sum_{i=1}^N \alpha_T(i)$$

# The Forward Trellis

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

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

$\alpha_{t-1}(i)$  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  $j$



1. Initialization:

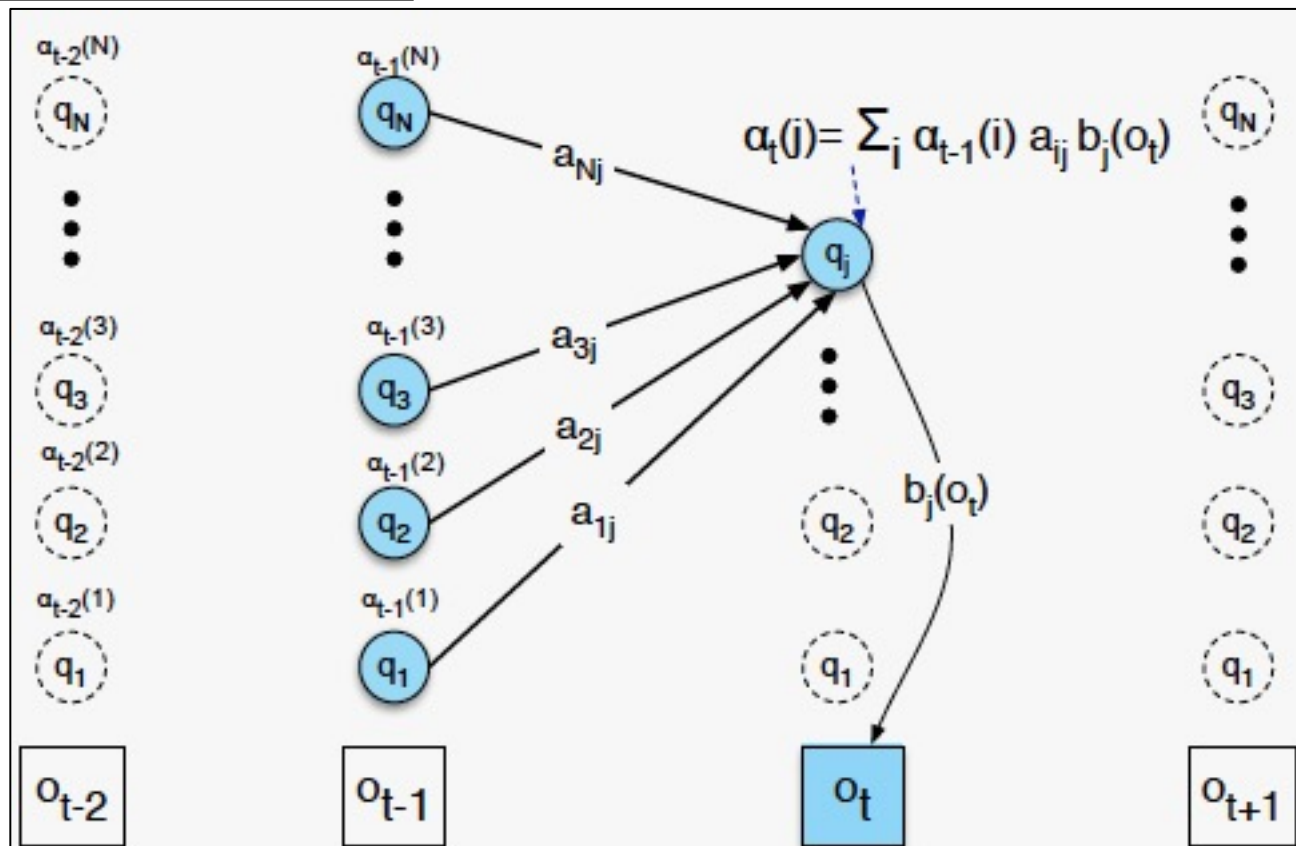
$$\alpha_1(j) = \pi_j b_j(o_1) \quad 1 \leq j \leq N$$

2. Recursion:

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

3. Termination:

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



# The Forward Algorithm

## proof of recursion:

always conditional \* prior

$$\begin{aligned}\alpha_t(j) &= P(q_t = j, o_{1:t} | \lambda) && \text{(definition)} \\ &= P(q_t = j, o_t, o_{1:t-1} | \lambda) && \text{(definition of } o_{1:t} \text{ notation)} \\ &= P(o_t | q_t = j, o_{1:t-1}, \lambda) P(q_t = j, o_{1:t-1} | \lambda) && \text{(simple decomposition of joint)} \\ &= P(o_t | q_t = j, \lambda) \sum_{i=1}^N P(q_t = j, q_{t-1} = i, o_{1:t-1} | \lambda) && \text{(first term: definition of HMM: observations only depend on state; second term: summing out variables)} \\ &= P(o_t | q_t = j, \lambda) \sum_{i=1}^N P(q_t = j | q_{t-1} = i, o_{1:t-1}, \lambda) P(q_{t-1} = i, o_{1:t-1} | \lambda) && \text{(simple decomposition of joint)} \\ &= P(o_t | q_t = j, \lambda) \sum_{i=1}^N P(q_t = j | q_{t-1} = i, \lambda) P(q_{t-1} = i, o_{1:t-1} | \lambda) && \text{(definition of HMM: state only depends on prev. state)} \\ &= b_j(o_t) \sum_{i=1}^N a_{ij} \alpha_{t-1}(i) && \text{(definition / notations)}\end{aligned}$$

# Using Forward Algorithm For Filtering

i.e. compute

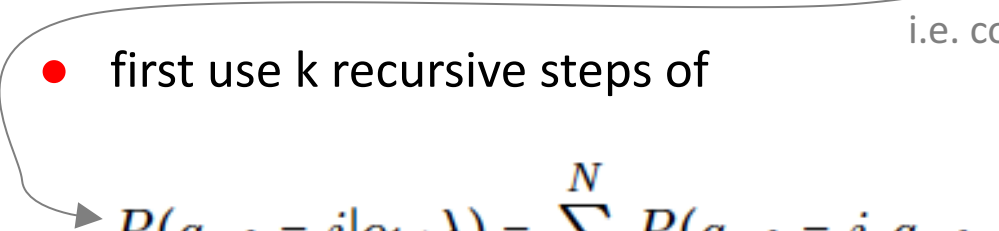
$$\begin{aligned}\tilde{\alpha}_t(j) &= P(q_t = j | o_{1:t} \lambda) \leftarrow \\ &= P(q_t = j | o_t, o_{1:t-1} \lambda) \\ &\propto P(o_t | q_t = j, o_{1:t-1}, \lambda) P(q_t = j | o_{1:t-1}, \lambda) \\ &= P(o_t | q_t = j, \lambda) \sum_{i=1}^N P(q_t = j, q_{t-1} = i | o_{1:t-1}, \lambda) \\ &= P(o_t | q_t = j, \lambda) \sum_{i=1}^N P(q_t = j | q_{t-1} = i, o_{1:t-1}, \lambda) P(q_{t-1} = i | o_{1:t-1}, \lambda) \\ &= P(o_t | q_t = j, \lambda) \sum_{i=1}^N P(q_t = j | q_{t-1} = i, \lambda) P(q_{t-1} = i | o_{1:t-1}, \lambda) \\ &= b_j(o_t) \sum_{i=1}^N a_{ij} \tilde{\alpha}_{t-1}(i)\end{aligned}$$

(application of Bayes rule  $\rightarrow$  we require normalization after each recursion)

# Using Forward Algorithm For Prediction

i.e. compute

- first use k recursive steps of


$$\begin{aligned}P(q_{t+k} = j | o_{1:t}, \lambda) &= \sum_{m=1}^N P(q_{t+k} = j, q_{t+k-1} = m | o_{1:t}, \lambda) \\&= \sum_{m=1}^N P(q_{t+k} = j | q_{t+k-1} = m, o_{1:t}, \lambda) P(q_{t+k-1} = m | o_{1:t}, \lambda) \\&= \sum_{m=1}^N P(q_{t+k} = j | q_{t+k-1} = m, \lambda) P(q_{t+k-1} = m | o_{1:t}, \lambda) \\&= \sum_{m=1}^N a_{mj} P(q_{t+k-1} = m | o_{1:t}, \lambda)\end{aligned}$$

to reduce to filtering problem, then proceed with filtering.

Naturally these k steps depend only on Markov chain (the  $a_{mj}$ ). If k is large: we will get the probability of state j in the stationary distribution of the Markov chain



# Decoding: The Viterbi Algorithm

- **Decoding**: Given HMM  $\lambda(A, B)$  and an observation sequence  $O = o_{1:T}$ , find the **most probable state sequence**  $Q = q_{1:T}$
- Again: blindly „trying out“ all possible sequences

$$Q_{max} = \operatorname{argmax}_Q P(Q|O) = \operatorname{argmax}_Q P(O|Q)P(Q)$$

is prohibitively expensive, because an exponential number of possible state sequences exist → again: recursive, dynamic programming approach: **Viterbi** algorithm

# Decoding: The Viterbi Algorithm

$$v_t(j) = \max_{q_{1:t-1}} P(q_{1:t-1}, q_t = j, o_{1:t} | \lambda)$$

## 1. Initialization:

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

## 2. Recursion

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

## 3. Termination:

$$\text{The best score: } P^* = \max_{i=1}^N v_T(i)$$

$$\text{The start of backtrace: } q_T^* = \operatorname{argmax}_{i=1}^N v_T(i)$$

simple step by step **backtrace**  $bt_t(j)$  (not to be confused with the backward message for smoothing) determines the most probable state in time-step  $t$  if most probable state in time-step  $t+1$  was  $q_{t+1} = j \rightarrow$

desired result:  $q_{1:T} = bt_1(bt_2(\dots bt_{T-1}(q_T^*) \dots)), bt_2(bt_3(\dots bt_{T-1}(q_T^*) \dots)), \dots, bt_{T-1}(q_T^*)$

# Decoding: The Viterbi Algorithm

both symbols are the same (both are “nu”) (just different fonts (Latex / Jurafksy mixed))

$$v_t(j) = \max_{q_{1:t-1}} P(q_{1:t-1}, q_t = j, o_{1:t} | \lambda)$$

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

## 2. Recursion

bt stand for backtrace, to create path from end to beginning. We are doing all staff to create a find most probable path

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

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

## 3. Termination:

$$\text{The best score: } P^* = \max_{i=1}^N v_T(i)$$

$$\text{The start of backtrace: } q_T^* = \operatorname{argmax}_{i=1}^N v_T(i)$$

simple step by step **backtrace**  $bt_t(j)$  (not to be confused with the backward message for smoothing) determines the most probable state in time-step  $t$  if most probable state in time-step  $t+1$  was  $q_{t+1} = j \rightarrow$

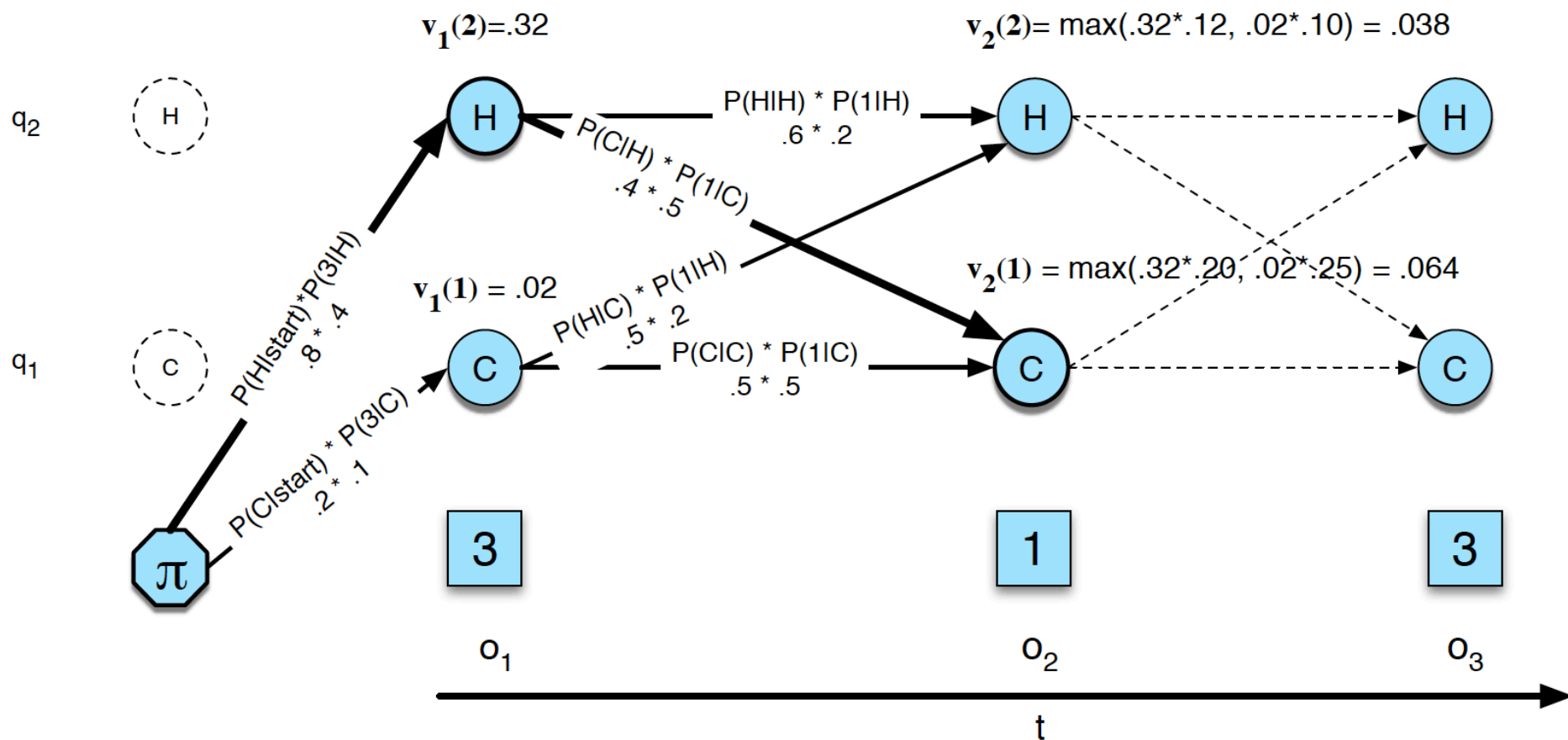
desired result:  $q_{1:T} = bt_1(bt_2(\dots bt_{T-1}(q_T^*) \dots)), bt_2(bt_3(\dots bt_{T-1}(q_T^*) \dots)), \dots, bt_{T-1}(q_T^*)$

# The Viterbi Trellis

$$v_t(j) = \max_{q_{1:t-1}} P(q_{1:t-1}, q_t = j, o_{1:t} | \lambda)$$

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

$v_{t-1}(i)$	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 $j$



# 1. Initialization:

$$v_1(j) = \pi_j b_j(o_1) \quad 1 \leq j \leq N$$

$$bt_1(j) = 0 \quad 1 \leq j \leq N$$

# 2. Recursion

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

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

# 3. Termination:

$$\text{The best score: } P^* = \max_{i=1}^N v_T(i)$$

$$\text{The start of backtrace: } q_T^* = \operatorname{argmax}_{i=1}^N v_T(i)$$

simple step by step **backtrace**  $bt_t(j)$

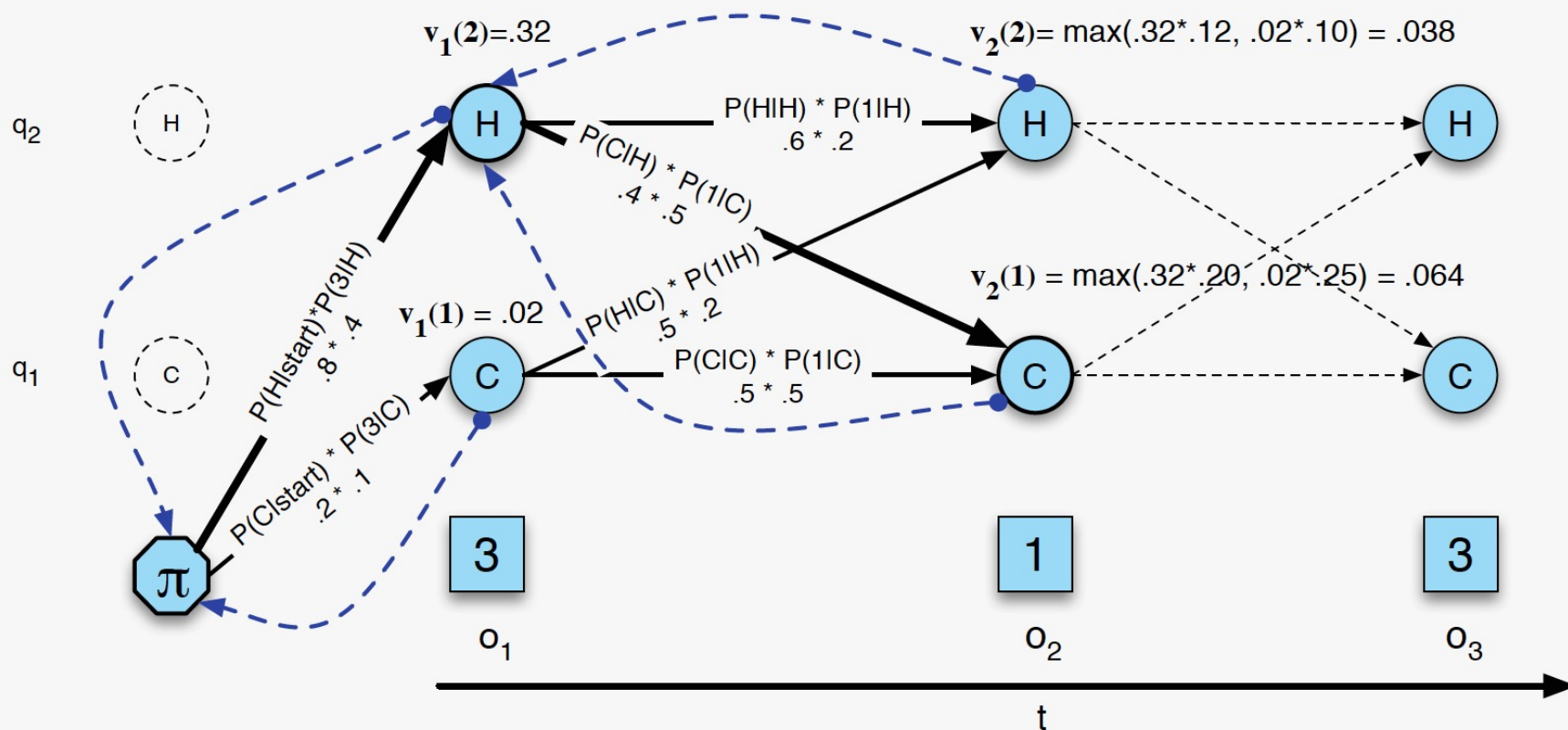
determines the most probable state in time-step t if most probable state in time-step t+1 was  $q_{t+1} = j$

→

desired result:

$$q_{1:T} = bt_1(bt_2(\dots bt_{T-1}(q_T^*) \dots)),$$

$$bt_2(bt_3(\dots bt_{T-1}(q_T^*) \dots)), \dots, bt_{T-1}(q_T^*)$$



# Decoding: The Viterbi Algorithm

## proof of recursion:

one could as well have started from  $P(q_{1:t-1}, q_t = j | o_{1:t} \lambda)$   
since  $o_{1:T}$  is observed and thus  $P(o_{1:t} | \lambda)$  is a constant.

$$\begin{aligned}
 v_t(j) &= \max_{q_{1:t-1}} P(q_{1:t-1}, q_t = j, o_{1:t} | \lambda) \\
 &= \max_{q_{1:t-1}} P(q_{1:t-1}, q_t = j, o_t, o_{1:t-1} | \lambda) \\
 &= \max_{q_{1:t-1}} P(o_t | q_t = j, q_{1:t-1}, o_{1:t-1}, \lambda) P(q_t = j, q_{1:t-1}, o_{1:t-1} | \lambda) \\
 &= \max_{q_{1:t-1}} P(o_t | q_t = j, \lambda) P(q_t = j, q_{1:t-1}, o_{1:t-1} | \lambda) \\
 &= \max_{q_{1:t-2}} \max_{i=1}^N P(o_t | q_t = j, \lambda) P(q_t = j, q_{t-1} = i, q_{1:t-2}, o_{1:t-1} | \lambda) \\
 &= P(o_t | q_t = j, \lambda) \max_{i=1}^N P(q_t = j | q_{t-1} = i, q_{1:t-2}, o_{1:t-1}, \lambda) \max_{q_{1:t-2}} P(q_{1:t-2}, q_{t-1} = i, o_{1:t-1} | \lambda) \\
 &= P(o_t | q_t = j, \lambda) \max_{i=1}^N P(q_t = j | q_{t-1} = i, \lambda) \max_{q_{1:t-2}} P(q_{1:t-2}, q_{t-1} = i, o_{1:t-1} | \lambda) \\
 &= b_j(o_t) \max_{i=1}^N a_{ij} v_{t-1}(i)
 \end{aligned}$$

(Definition of HMM (Markov Assumption of observation model): observations only depend on the state in the respective time step)

making max explicit for t-1 state

Definition of HMM (Markov Assumption for state transition model)

(definition / notations)

→ essentially the same as for Forward, just replace sum with max

# Forward-Backward (Baum Welch) Algorithm

- Goal: given structure of HMM (set of states) and an observation sequence  $O = o_{1:T}$ , **learn** the state transition model A and the observation (emission) model B (essentially via MLE with **EM-algorithm**)
- for a **simple Markov model** we know the state sequence, so the MLE for A is given by **counting**:

$$a_{ij} = \frac{C(i \rightarrow j)}{\sum_{q \in Q} C(i \rightarrow q)}$$

but for an **HMM**, the states are **hidden** variables.

- **Idea**: **combine Forward** algorithm (recursively passing a forward message  $\alpha_t(j)$  " $\rightarrow$ ") with **Backward** algorithm (recursively passing a backward message  $\beta_t(i)$  " $\leftarrow$ ") and **iterate** (EM style)

$\beta_t(i) = P(o_{t+1}, o_{t+2} \dots o_T | q_t = i, \lambda)$  is the probability for future observations, assuming to be now in state  $i$

( compare forward message  $\alpha_t(j) = P(o_1, o_2 \dots o_t, q_t = j | \lambda)$  )

## 1. Initialization:

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

## 2. Recursion

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

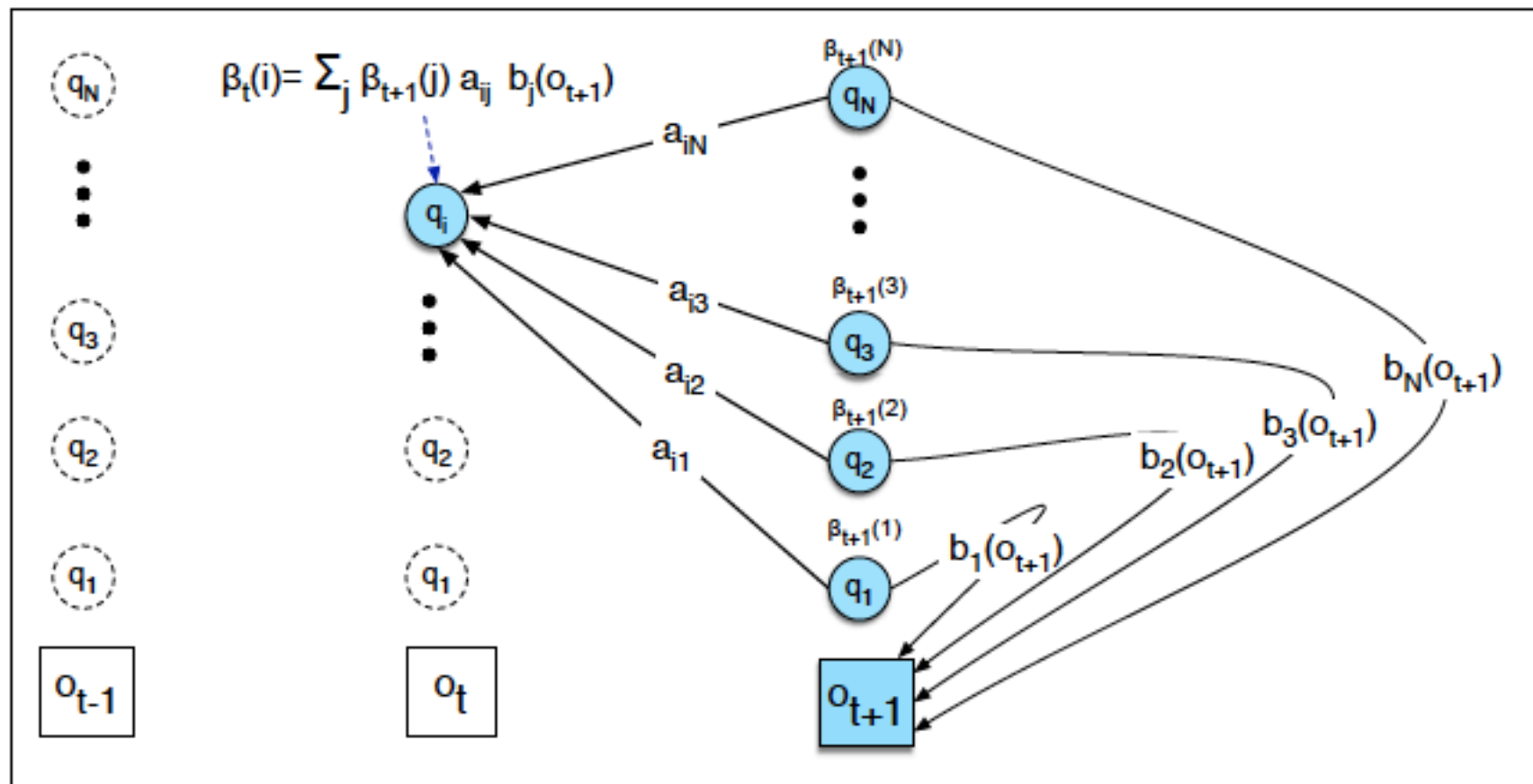
## 3. Termination:

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



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

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



## proof of recursion

$$\begin{aligned}\beta_t(i) &= P(o_{t+1:T} | q_t = i, \lambda) \\&= \sum_{j=1}^N P(o_{t+1:T}, q_{t+1} = j | q_t = i, \lambda) \\&= \sum_{j=1}^N P(o_{t+1:T} | q_{t+1} = j, q_t = i, \lambda) P(q_{t+1} = j | q_t = i, \lambda) \\&= \sum_{j=1}^N P(o_{t+1:T} | q_{t+1} = j, \lambda) P(q_{t+1} = j | q_t = i, \lambda) \\&= \sum_{j=1}^N P(o_{t+1}, o_{t+2:T} | q_{t+1} = j, \lambda) P(q_{t+1} = j | q_t = i, \lambda) \\&= \sum_{j=1}^N P(o_{t+1} | o_{t+2:T}, q_{t+1} = j, \lambda) P(o_{t+2:T} | q_{t+1} = j, \lambda) P(q_{t+1} = j | q_t = i, \lambda) \\&= \sum_{j=1}^N P(o_{t+1} | q_{t+1} = j, \lambda) P(o_{t+2:T} | q_{t+1} = j, \lambda) P(q_{t+1} = j | q_t = i, \lambda) \\&= \sum_{j=1}^N b_j(o_{t+1}) \beta_{t+1}(j) a_{ij}\end{aligned}$$

# Forward – Backward Algorithm

goal: iteratively improve estimates for  $a_{ij}$  and  $b_i(o_t)$

1. estimate for  $a_{ij}$  :

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

Define:

$$\begin{aligned}\xi_t(i, j) &= P(q_t = i, q_{t+1} = j | O, \lambda) \\ &= \frac{P(q_t = i, q_{t+1} = j, O | \lambda)}{P(O | \lambda)} \\ &= \frac{\alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)}{\sum_{j=1}^N \alpha_t(j) \beta_t(j)}\end{aligned}$$

since

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

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

# Forward – Backward Algorithm

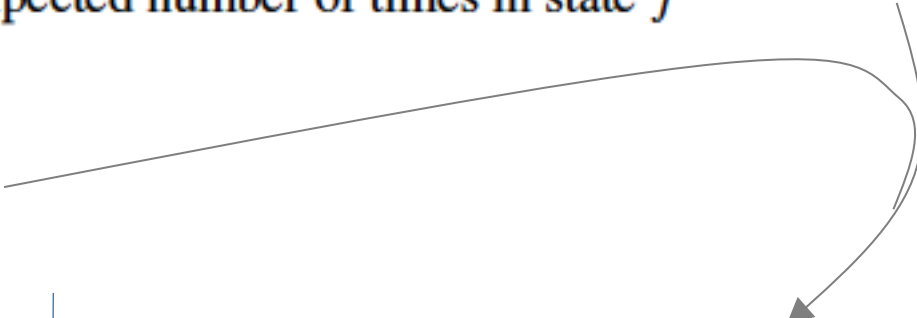
goal: iteratively improve estimates for  $a_{ij}$  and  $b_i(o_t)$

2. estimate for  $b_i(o_t)$

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

Define:

$$\begin{aligned}\gamma_t(j) &= P(q_t = j | O, \lambda) \\ &= \frac{P(q_t = j, O | \lambda)}{P(O | \lambda)} \\ &= \frac{\alpha_t(j) \beta_t(j)}{P(O | \lambda)} \\ &= \frac{\alpha_t(j) \beta_t(j)}{\sum_{j=1}^N \alpha_t(j) \beta_t(j)}\end{aligned}$$


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

# Forward – Backward Algorithm

**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**

$$\gamma_t(j) = \frac{\alpha_t(j)\beta_t(j)}{\sum_{j=1}^N \alpha_t(j)\beta_t(j)} \quad \forall t \text{ and } j$$

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

**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**  $A, B$

# Forward – Backward Algorithm

## proof of recursion for $\xi$

$$\begin{aligned}\widehat{\xi}_t(i, j) &= P(q_t = i, q_{t+1} = j, o_{1:T} | \lambda) && \text{definition} \\ &= P(q_t = i, q_{t+1} = j, o_{1:t}, o_{t+1}, o_{t+2:T} | \lambda) \\ &= P(o_{t+1} | q_t = i, q_{t+1} = j, o_{1:t}, o_{t+2:T}, \lambda) P(q_t = i, q_{t+1} = j, o_{1:t}, o_{t+2:T} | \lambda) \\ &= P(o_{t+1} | q_t = i, q_{t+1} = j, o_{1:t}, o_{t+2:T}, \lambda) P(o_{t+2:T} | q_t = i, q_{t+1} = j, o_{1:t}, \lambda) P(q_t = i, q_{t+1} = j, o_{1:t} | \lambda) \\ &= P(o_{t+1} | q_t = i, q_{t+1} = j, o_{1:t}, o_{t+2:T}, \lambda) P(o_{t+2:T} | q_t = i, q_{t+1} = j, o_{1:t}, \lambda) P(q_{t+1} = j | q_t = i, o_{1:t}, \lambda) \\ &\quad P(q_t = i, o_{1:t} | \lambda) \\ &= P(o_{t+1} | q_{t+1} = j, \lambda) P(o_{t+2:T} | q_{t+1} = j, \lambda) P(q_{t+1} = j | q_t = i, \lambda) P(q_t = i, o_{1:t} | \lambda) \\ &= b_j(o_{t+1}) \beta_{t+1}(j) a_{ij} \alpha_t(i) && \text{definitions}\end{aligned}$$

straight-forward chain decomposition of joint

definition of HMM:

- (1) Markov assumption for observation model
- (2)  $\rightarrow$  observations are conditionally independent of states further in the past, given states in the past temporally closer to the observation time AND conditionally independent of observations of the past given states in the past temporally closer to the observation time
- (3) Markov assumption for transition model

# Forward – Backward Algorithm

## proof of recursion for $\gamma$

$$\begin{aligned}\gamma_t(j) &= P(q_t = j, o_{1:T} | \lambda) && \text{definition} \\ &= P(q_t = j, o_{1:t}, o_{t+1:T} | \lambda) \\ &= P(o_{t+1:T} | q_t = j, o_{1:t}, \lambda) P(q_t = j, o_{1:t} | \lambda) \\ &= P(o_{t+1:T} | q_t = j, \lambda) P(q_t = j, o_{1:t} | \lambda) \\ &= \beta_t(j) \alpha_t(j) && \text{definitions}\end{aligned}$$

defintion of HMM: conditionally independent of observations of the past  
given states in the past temporally closer or as close to the observation time

# Forward – Backward Algorithm

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

$$P(O|\lambda) = \sum_{j=1}^N P(o_{1:T}, q_T = j | \lambda) = \sum_{j=1}^N \alpha_T(j) \quad \text{via T forward-messages}$$

$$\begin{aligned} P(O|\lambda) &= \sum_{j=1}^N P(o_{1:T}, q_1 = j | \lambda) \\ &= \sum_{j=1}^N P(q_1 = j | \lambda) P(o_1, o_{2:T} | q_1 = j, \lambda) \\ &= \sum_{j=1}^N P(q_1 = j | \lambda) P(o_1 | o_{2:T}, q_1 = j, \lambda) P(o_{2:T} | q_1 = j, \lambda) \\ &= \sum_{j=1}^N P(q_1 = j | \lambda) P(o_1 | q_1 = j, \lambda) P(o_{2:T} | q_1 = j, \lambda) \\ &= \sum_{j=1}^N \pi_j b_j(o_1) \beta_1(j) \quad \text{via T-1 backward-messages} \end{aligned}$$



# Forward – Backward Algorithm

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

$$P(O|\lambda) =$$

$$= \sum_{j=1}^N P(o_{1:T}, q_t = j | \lambda)$$

$$= \sum_{j=1}^N P(o_{1:t}, o_{t+1:T}, q_t = j | \lambda)$$

$$= \sum_{j=1}^N P(o_{1:t}, o_{t+1:T} | q_t = j, \lambda) P(q_t = j | \lambda)$$

$$= \sum_{j=1}^N P(o_{1:t} | q_t = j, \lambda) P(o_{t+1:T} | q_t = j, \lambda) P(q_t = j | \lambda)$$

$$= \sum_{j=1}^N P(o_{1:t}, q_t = j | \lambda) P(o_{t+1:T} | q_t = j, \lambda)$$

$$= \sum_{j=1}^N \alpha_t(j) \beta_t(j)$$

via  $t$  forward-messages  
and  $T-(t-1)$  backward messages



- (1) Dan Jurafsky and James Martin: Speech and Language Processing (3<sup>rd</sup> ed. draft, version Oct 2019); Online: <https://web.stanford.edu/~jurafsky/slp3/> (URL, Oct 2019) (this slide-set is especially related to chapter Appendix A)
- (2) S. Russell, P. Norvig: Artificial Intelligence – A Modern Approach, 3<sup>rd</sup> edition, Pearson 2010, section 15.2

# Recommendations for Studying

- minimal approach:

work with the slides and understand their contents! Think beyond instead of merely memorizing the contents

- standard approach:

minimal approach + read the corresponding pages in Jurafsky [1]

- interested students

== standard approach