HMM Training: The Forward-Backward Algorithm

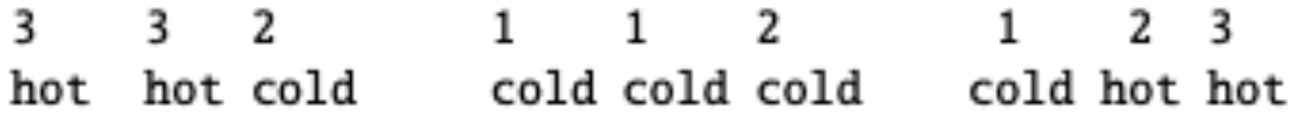
- Question Description:

  Given an observation sequence O and the set of possible states in the HMM, learn the HMM parameters A and B.

- The input to such a learning algorithm would be an unlabeled sequence of observations O and a vocabulary of potential hidden states Q.

Toy Example

- Let us begin by considering the much simpler case of training a fully visible Markov model, we’re know both the temperature and the ice cream count for every day. That is, imagine we see the following set of input observations and magically knew the aligned hidden state sequences：



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

Toy Example

- This would easily allow us to compute the HMM parameters just by maximum likelihood estimation from the training data. First, we can compute π from the count of the 3 initial hidden states:

$$\pi_h = 1/3 \quad \pi_c = 2/3$$

- Next we can directly compute the A matrix from the transitions, ignoring the final hidden states:

$$\begin{aligned} p(hot|hot) &= 2/3 \quad & p(cold|hot) &= 1/3 \\ p(cold|cold) &= 2/3 \quad & p(hot|cold) &= 1/3 \end{aligned}$$

Toy Example

- and the B matrix:

$$\begin{array}{rl}
P(1|hot) = 0/4 = 0 & p(1|cold) = 3/5 = .6 \\
P(2|hot) = 1/4 = .25 & p(2|cold = 2/5 = .4 \\
P(3|hot) = 3/4 = .75 & p(3|cold) = 0
\end{array}$$

- Note: For a real HMM, we cannot compute these counts directly from an observation sequence since we don’t know which path of states was taken through the machine for a given input.

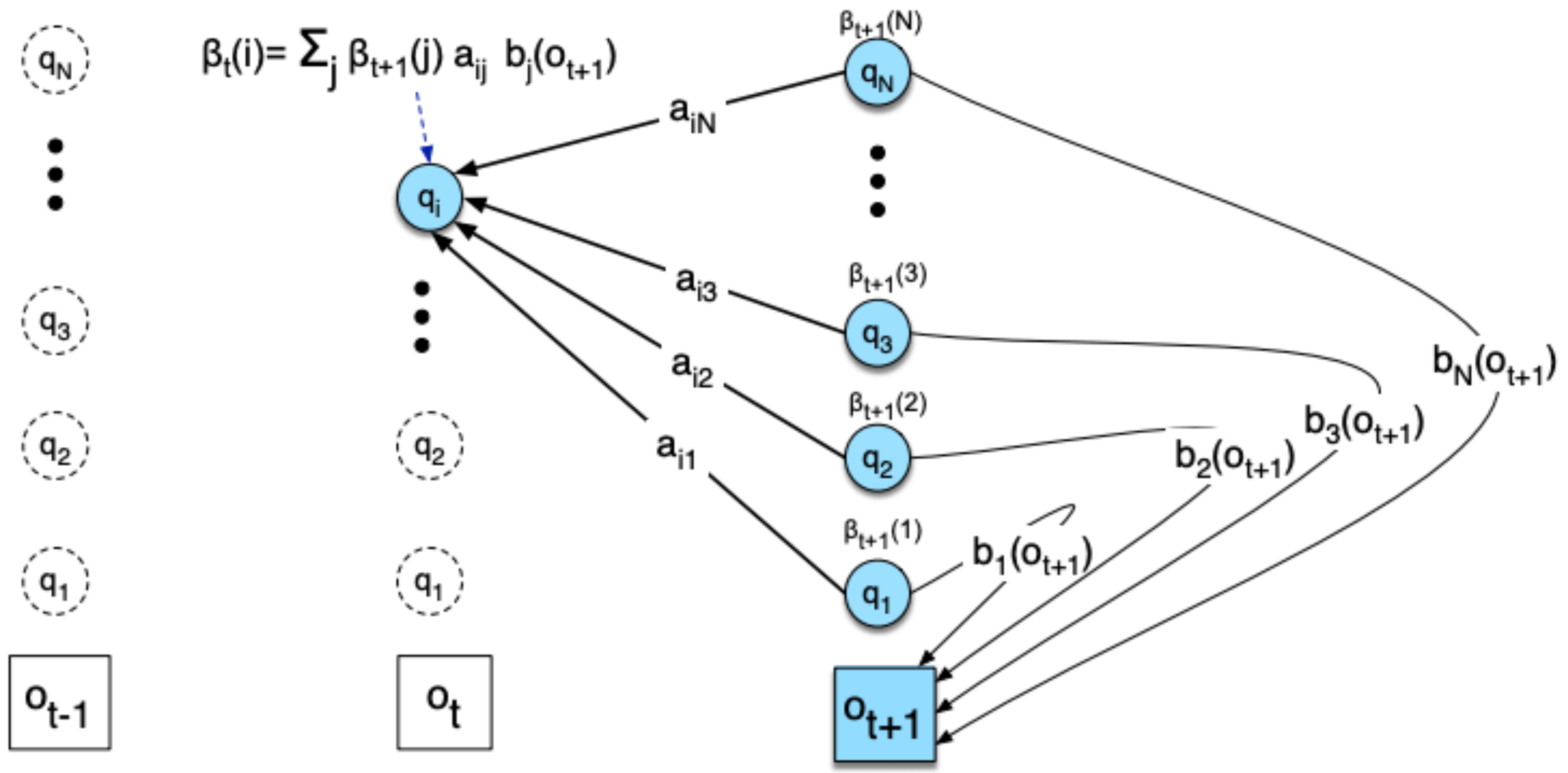
Backward Probability

- The backward probability β is the probability of seeing the observations from time t +1 to the end, given that we are in state i at time t (and given the automaton λ):

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

- It is computed inductively in a similar manner to the forward algorithm.

Computing Backward Probability



Computing Backward Probability

- Initialization

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

- Recursion

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

- Termination

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

The Forward-Backward Algorithm

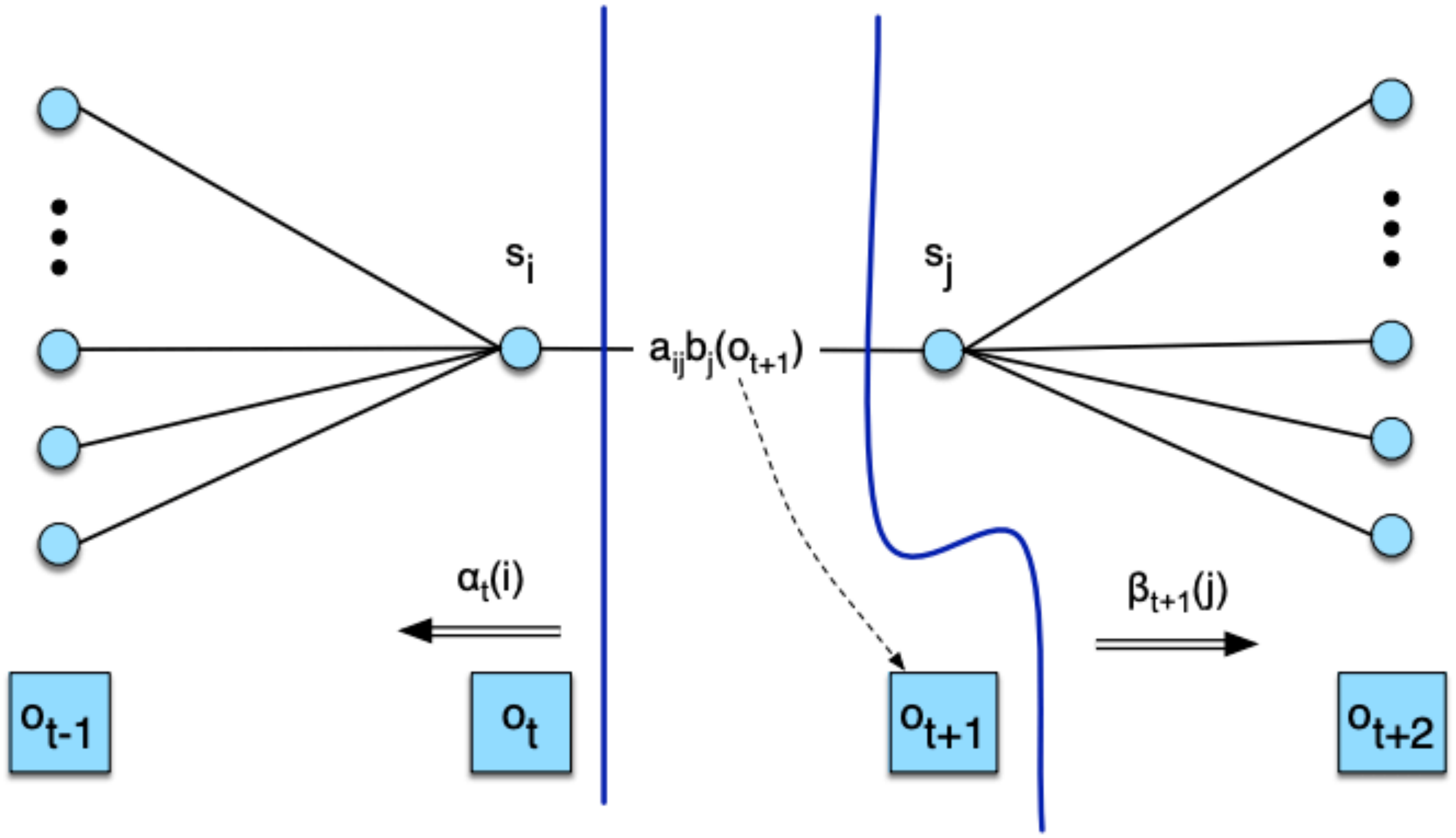
- We define the probability ξt as the probability of being in state i at time t and state j at time t +1, given the observation sequence and of course the model:

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

- To compute ξt , we first compute a probability which is similar to ξt ,

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

The Forward-Backward Algorithm



The Forward-Backward Algorithm

- The figure shows the various probabilities that go into computing not-quite-ξt : 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-ξt as follows:

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

The Forward-Backward Algorithm

- The probability of the observation given the model is simply the forward probability of the whole utterance:

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

- So, the final equation for ξt is

$$\xi_t(i,j) = \frac{\alpha_t(i)\, a_{ij} b_j(o_{t+1})\beta_{t+1}(j)}{\sum_{j=1}^{N} \alpha_t(j)\beta_t(j)} \qquad P(X|Y,Z) = \frac{P(X,Y|Z)}{P(Y|Z)}$$

The Forward-Backward Algorithm

- Let’s how to estimate aij by a variant of simple maximum likelihood estimation:

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

- Here’s the final formula:

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

The Forward-Backward Algorithm

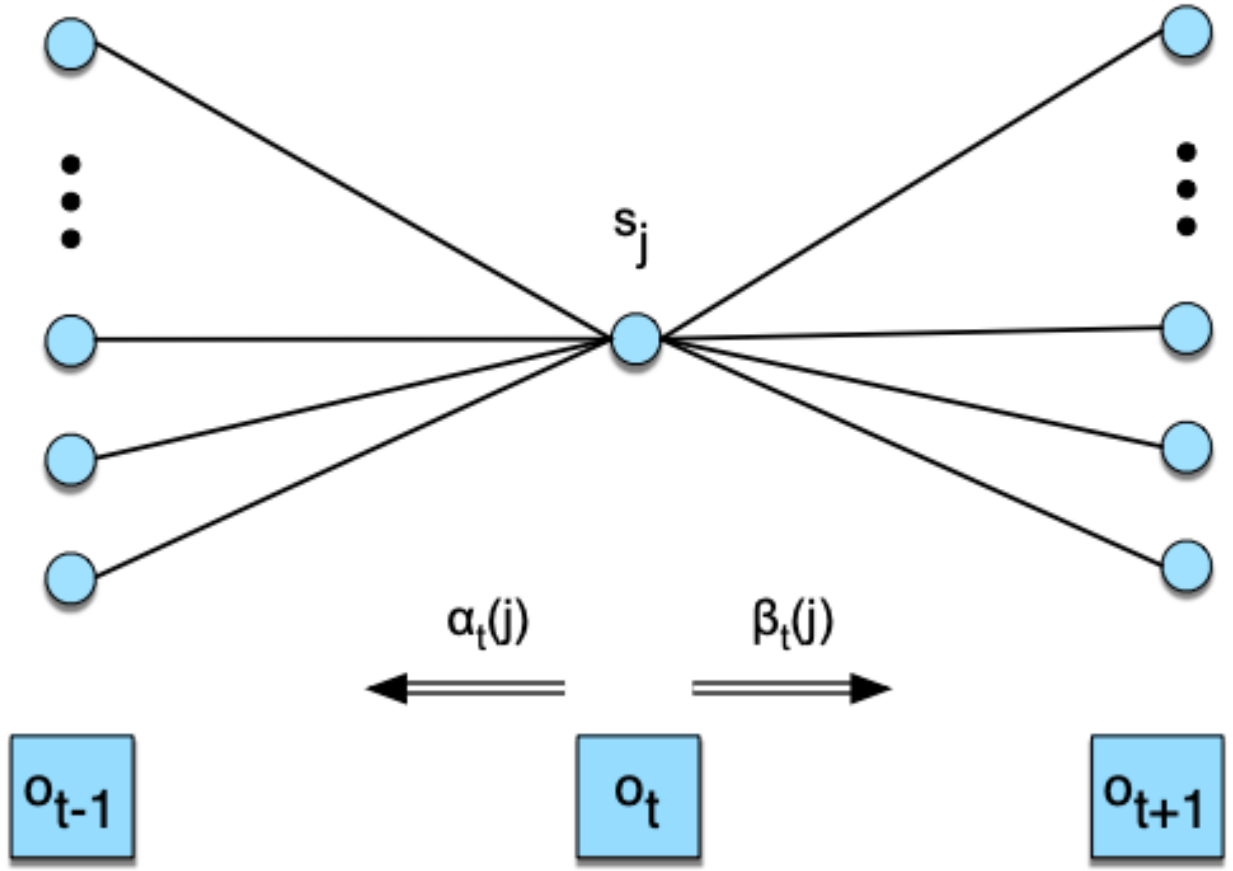
- For this, we will need to know the probability of being in state j at time t, which we will call γt(j):

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

- Once again, we will compute this by including the observation sequence in the probability:

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

The Forward-Backward Algorithm



The Forward-Backward Algorithm

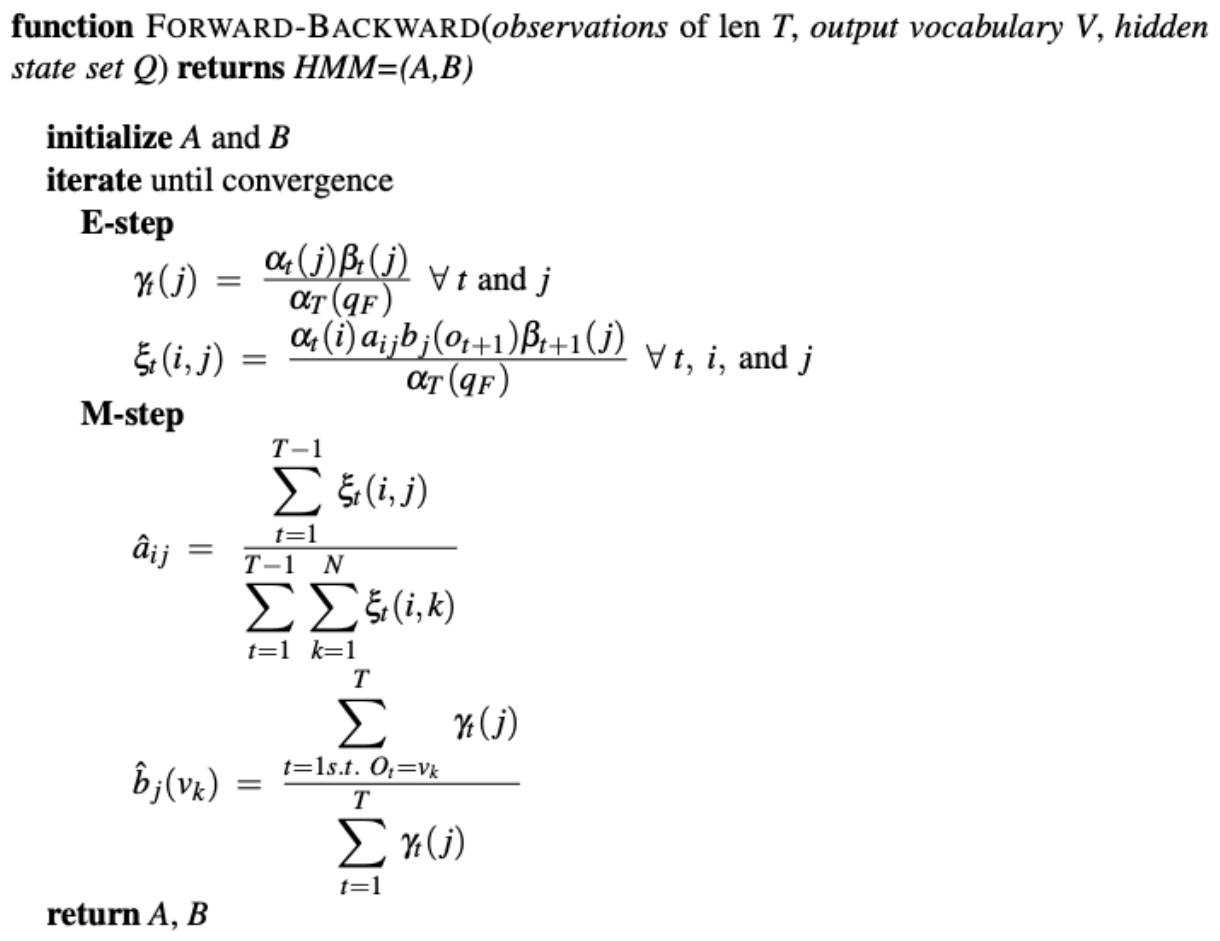
- the numerator is just the product of the forward probability and the backward probability:

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

- The result of b is the percentage of the times that we were in state j and saw symbol vk:

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

The Forward-Backward Algorithm



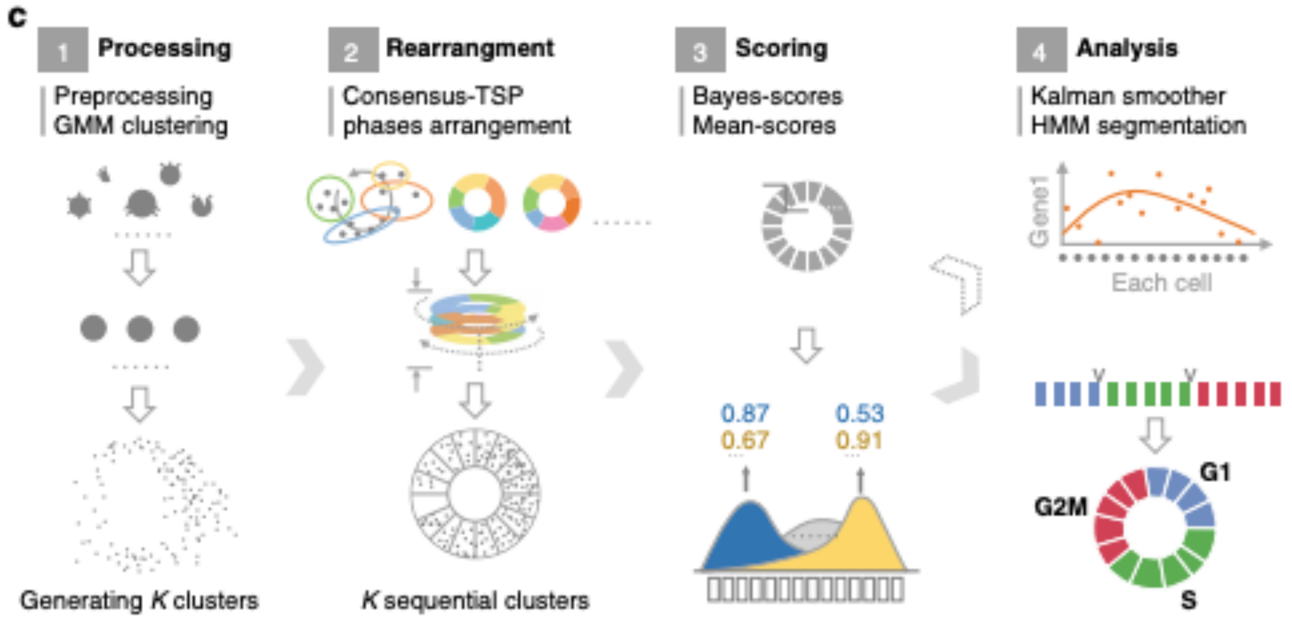
```
function FORWARD-BACKWARD(observations of len T, output vocabulary V, hidden
state set Q) returns HMM=(A,B)

  initialize A and B
  iterate until convergence
    E-step
      $\gamma_t(j) = \frac{\alpha_t(j)\beta_t(j)}{\alpha_T(q_F)} \quad \forall\, t \text{ and } j$
      $\xi_t(i,j) = \frac{\alpha_t(i)\, a_{ij} b_j(o_{t+1})\beta_{t+1}(j)}{\alpha_T(q_F)} \quad \forall\, t,\ i, \text{ and } j$
    M-step
      $\hat{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \sum_{k=1}^{N} \xi_t(i,k)}$
      $\hat{b}_j(v_k) = \frac{\sum_{t=1 s.t.\ O_t=v_k}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)}$
  return A, B
```

Summary

- Hidden Markov models (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.

Application

- reCAT


c
1 Processing
Preprocessing
GMM clustering
Generating K clusters
2 Rearrangment
Consensus-TSP
phases arrangement
K sequential clusters
3 Scoring
Bayes-scores
Mean-scores
0.87
0.67
0.53
0.91
4 Analysis
Kalman smoother
HMM segmentation
Gene1
Each cell
G1
G2M
S