ELEC-E5510 Speech Recognition

Hidden Markov Models

Tamás Grósz

Department of Signal Processing and Acoustics

HMM

HMM notations

- λ: the HMM model
- $O = \{o_1, o_2, \ldots, o_T\}$: a sequence of observations (e.g. MFCC features)
- $Q = \{q_1, q_2, \ldots, q_N\}$: a set of N (hidden) states, or a sequence of hidden states (e.g. state = phonemes)
- $A = \{a_{11}, \ldots, a_{ij}, \ldots, a_{NN}\}$: transition probability matrix
- $B = \{b_i(o_t)\}$: a sequence of observation likelihoods, also called emission probabilities (probability of an observation o_t being generated from a state i)
- $\pi = \{\pi_1, \ldots, \pi_N\}$: initial probability distribution

GMM-HMM parameters

How can we model $b_i(o_t)$? With **GMM**!

GMM

$$
\begin{aligned}
p(x) &= \sum_{m=1}^{M} w_m N_m(x|\mu_m, \Sigma_m) \\
&= \sum_{m=1}^{M} \frac{w_m}{\sqrt{(2\pi)^M |\Sigma_m|}} exp\left(-\frac{1}{2}(x-\mu_m)^T \Sigma_m^{-1}(x-\mu_m)\right)
\end{aligned}
$$

$$
\sum_{m=1}^{M} w_m = 1
$$

HMM assumptions

Markov assumption

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

The probability of a particular state depends only on the previous state.

Output independence

$P(o_t|Q, O) = P(o_t|q_t)$

The probability of an output observation o_t depends only on the state that produced the observation (q_t) and not on any other states or any other observations

HMM operations

Scoring

How to compute the probability of the observation sequence for a model?

Decoding

How to compute the best state sequence for the observations?

Training

How to set the model parameters to maximize the probability of the training samples?

Scoring

Scoring

- Given an observation sequence $O = \{o_1, o_2, \dots, o_T\}$
- What is the probability of generating it? $P(O|\lambda) =?$
- Assume that we know $Q = \{q_1, q_2, \dots, q_T\}$
- Then $P(O|Q, \lambda) = \prod_{t=1}^{T} P(o_t|q_t, \lambda)$
- And $P(o_t|q_t, \lambda) = b_{q_t}(o_t)$
- We still need $P(Q|\lambda) = \pi_{q_1} * a_{q_1 q_2} * a_{q_2 q_3} * \cdots * a_{q_{T-1} q_T}$
- Now we can rewrite $P(O|\lambda) = P(O|Q, \lambda)P(Q|\lambda)$ (chain rule)

Scoring equation

$$
\begin{aligned}
P(O|\lambda) &= \sum_{Q} P(O|q_t, \lambda)P(Q|\lambda) \\
&= \sum_{Q} \pi_{q_1} * b_{q_1}(o_1) * a_{q_1 q_2} b_{q_2}(o_2) * \cdots * a_{q_{T-1} q_T} b_{q_T}(o_T)
\end{aligned}
$$

Scoring

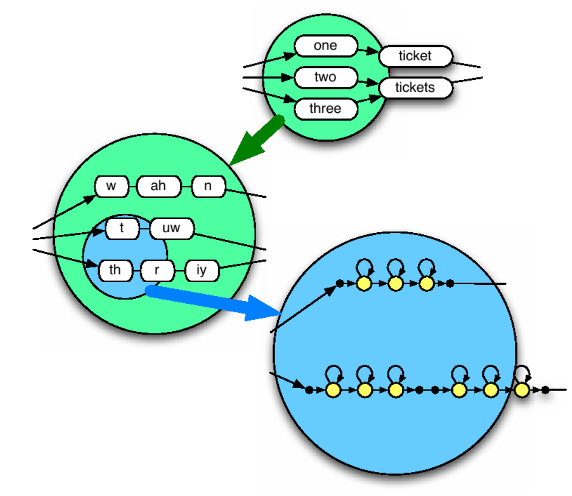
Scoring equation

$$P(O|\lambda) = \quad = \sum_{Q} \pi_{q_1} * b_{q_1}(o_1) * a_{q_1 q_2} b_{q_2}(o_2) * \cdots * a_{q_{T-1} q_T} b_{q_T}(o_T)$$

- Now we have the scoring method, but is it practical?
- NO
- It is not feasible to consider all possible state sequences separately (for N states and T observation we have $O(2T * N^T)$ sequences)
- We need a better way of handling the state sequences.

Scoring using a search network

- Let's create a search graph!
- Map words into phonemes and states.
- Using these mappings we can construct a search graph.



(Picture by S.Renals)

Forward algorithm

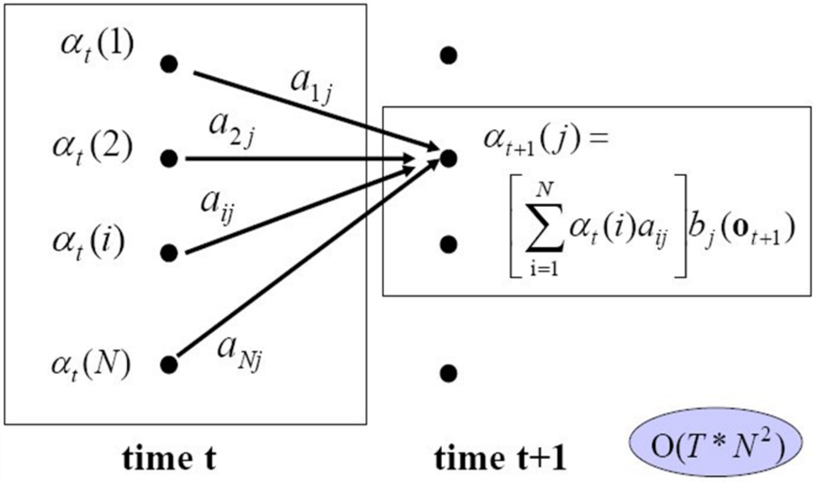
The goal of the Forward algorithm is to calculate the probability of observing $o_1, o_2, \ldots, o_t$ given an HMM (λ)

To achieve this we need to define the forward variable:
$\alpha_t(i) = P(o_1, o_2, \ldots, o_t, q_t = i|\lambda)$

How can we calculate $\alpha_t(i)$?

- Initialization: $\alpha_0(i) = \pi_i$ (non-emitting initial state)
- Induction: $\alpha_{t+1}(i) = \left[\sum_{j=1}^{N} \alpha_t(j) * \underbrace{a_{ji}}_{\text{Transition prob.}} \right] * \underbrace{b_i(o_{t+1})}_{\text{observation prob.}}$
- Termination: $P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i)$

Forward algorithm



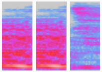
Picture by B. Pellom

Forward algorithm, exercise 1

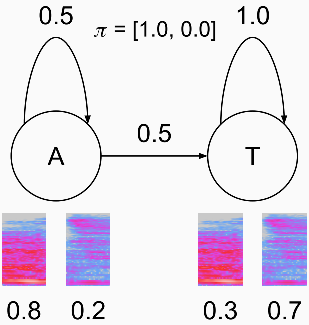
Given an HMM, and the initial probabilities:

$\pi = [\underbrace{1.0}_{A}, \underbrace{0.0}_{T}]$

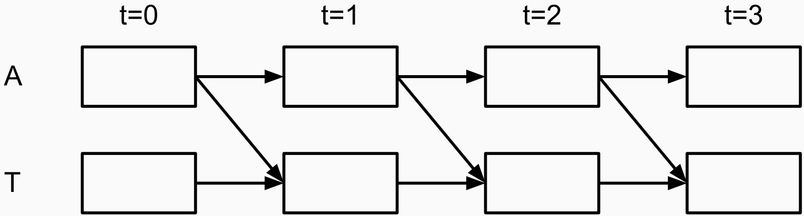
What is the probability of observing

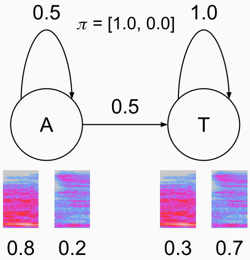
$O =$ 

$P(O|\lambda) = ?$



Forward algorithm, exercise 1 solution

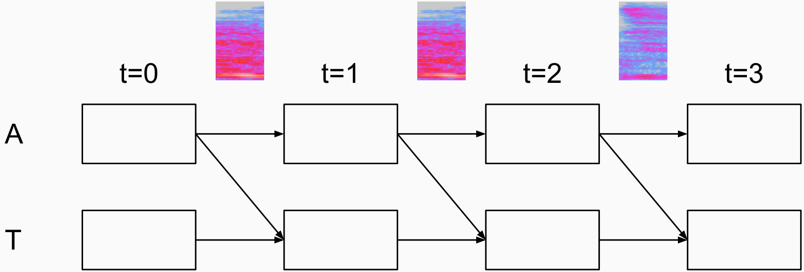


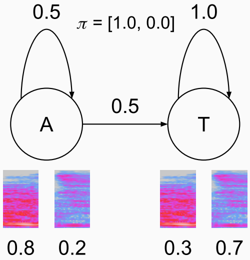


Answer

$P(O|\lambda) =$

Forward algorithm, exercise 1 solution

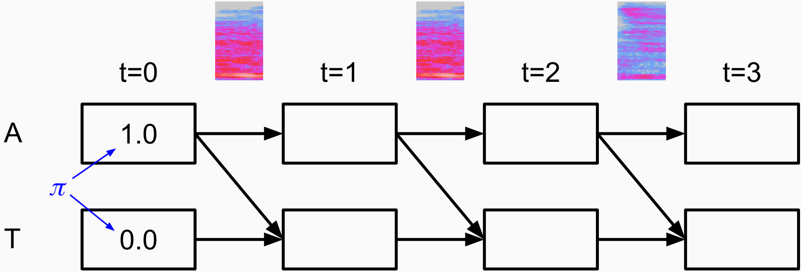


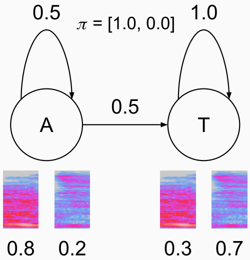


Answer

$P(O|\lambda) =$

Forward algorithm, exercise 1 solution

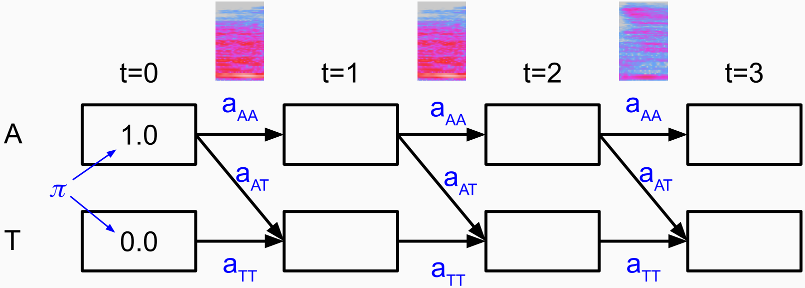


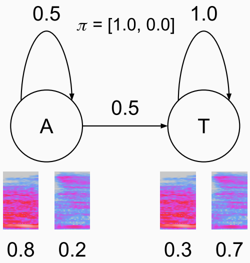


Answer

$P(O|\lambda) =$

Forward algorithm, exercise 1 solution

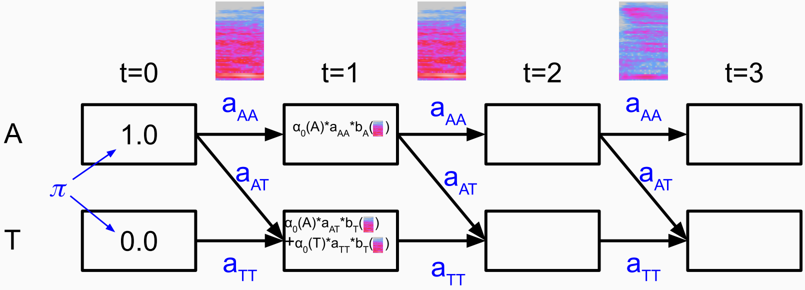


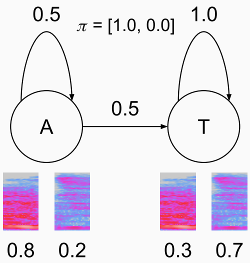


Answer

$P(O|\lambda) =$

Forward algorithm, exercise 1 solution

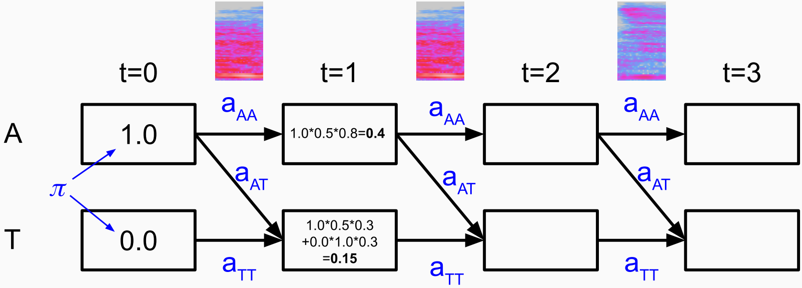


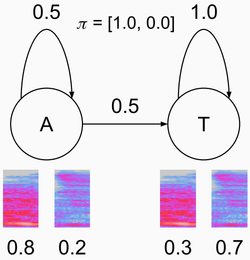


Answer

$P(O|\lambda) =$

Forward algorithm, exercise 1 solution

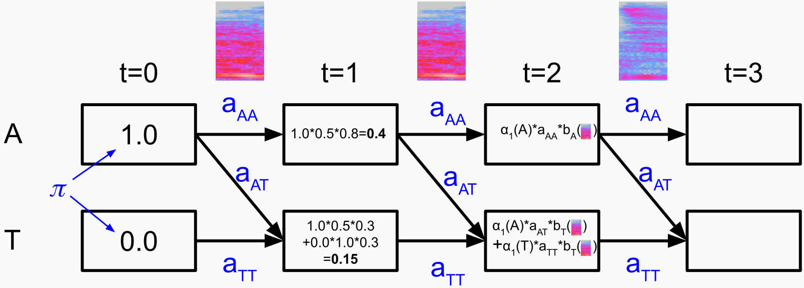


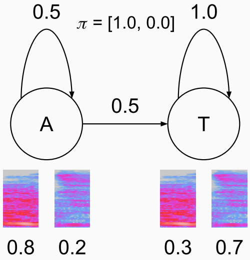


Answer

$P(O|\lambda) =$

Forward algorithm, exercise 1 solution

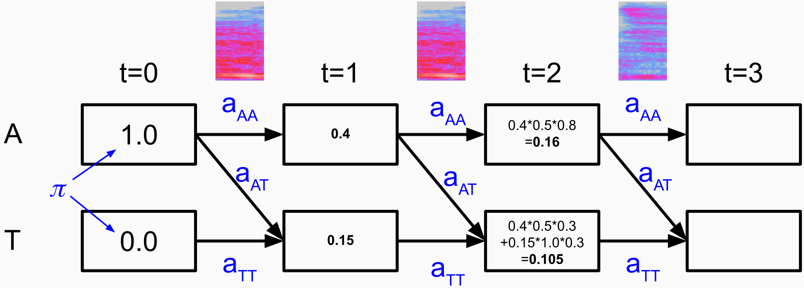


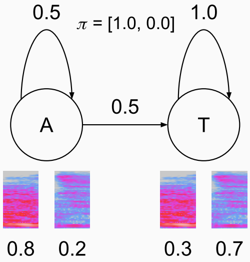


Answer

$P(O|\lambda) =$

Forward algorithm, exercise 1 solution

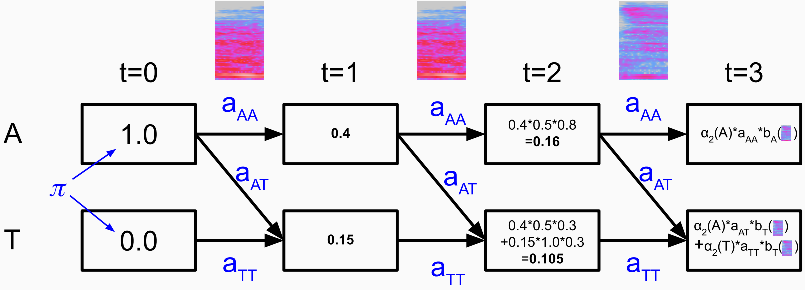


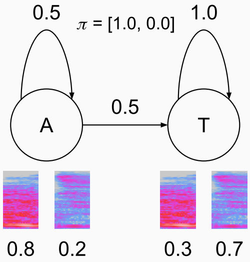


Answer

$P(O|\lambda) =$

Forward algorithm, exercise 1 solution

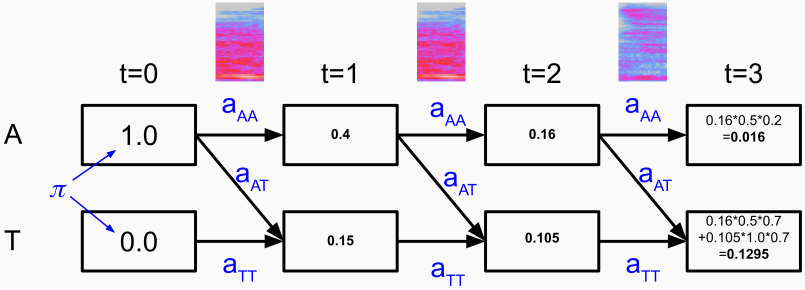


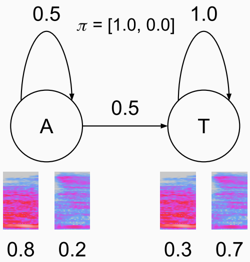


Answer

$P(O|\lambda) =$

Forward algorithm, exercise 1 solution

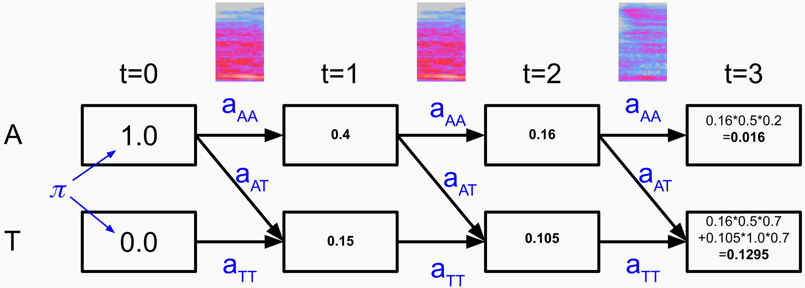


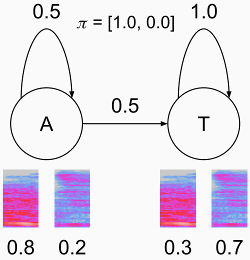


Answer

$P(O|\lambda) =$

Forward algorithm, exercise 1 solution





Answer

$P(O|\lambda) =$

$\sum_{i=1}^{N} \alpha_T(i) = 0.016 + 0.1295 = 0.1455$

Decoding

Decoding

- Given a sequence of observations $O = o_1, o_2, \ldots, o_T$
- What is the sequence of hidden states $Q = q_1, q_2, \ldots, q_T = ?$
- That maximizes $P(O, Q|\lambda)$

Viterbi algorithm

Initialization

$$\delta_1(i) = \pi_i b_i(o_1) \text{ and } \psi_1(i) = 0$$

Recursion

$$\delta_t(i) = \max_{1 \leq j \leq N} [\delta_{t-1}(j) a_{ji}] b_i(o_t)$$

$$\psi_t(i) = \operatorname*{argmax}_{1 \leq j \leq N} [\delta_{t-1}(j) a_{ji}]$$

Termination

$$P^* = \max_{1 \leq i \leq N} \delta_T(i)$$

$$q_T^* = \operatorname*{argmax}_{1 \leq i \leq N} \delta_T(i)$$

Backtrace

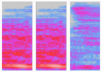
$$q_t^* = \psi_{t+1}(q_{t+1}^*)$$

Viterbi algorithm, exercise 2

Given an HMM, the initial probabilities:

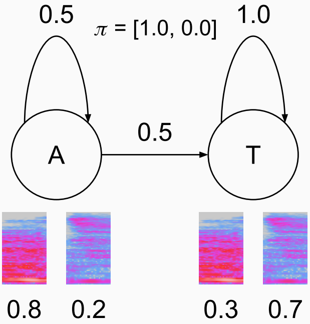
$\pi = [\underbrace{1.0}_{A}, \underbrace{0.0}_{T}]$

And the observation sequence

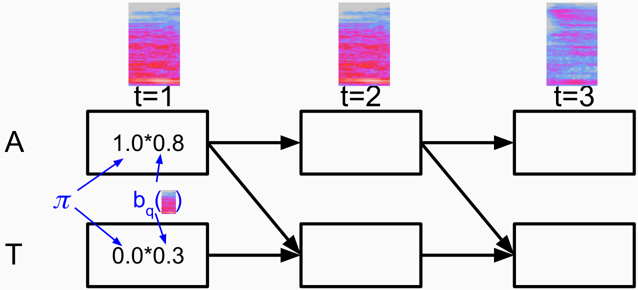
$O =$ 

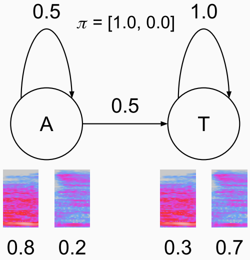
What is the most probable state-sequence (Q)?

$argmax_Q P(O, Q|\lambda) =?$



Viterbi algorithm, exercise 2 solution

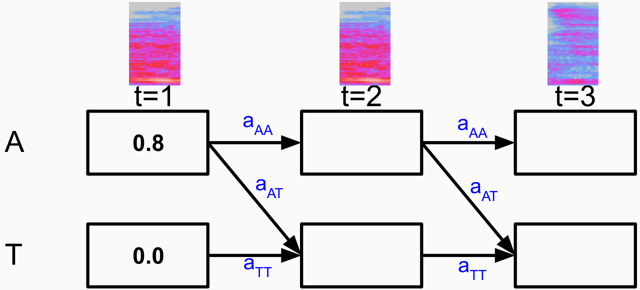


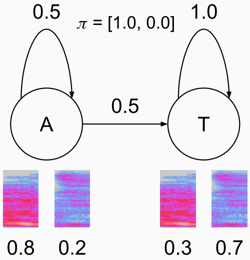


Answer

$Q =$

Viterbi algorithm, exercise 2 solution

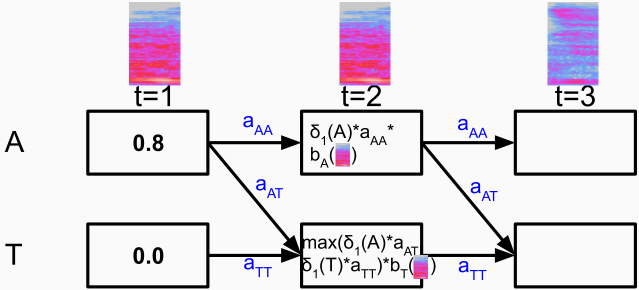


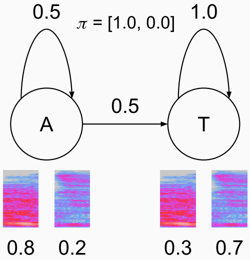


Answer

$Q =$

Viterbi algorithm, exercise 2 solution

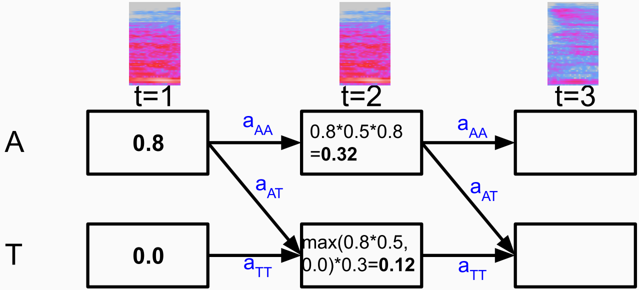


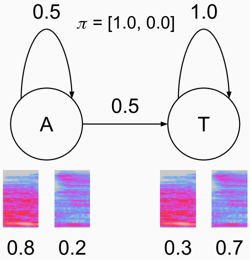


Answer

$Q =$

Viterbi algorithm, exercise 2 solution

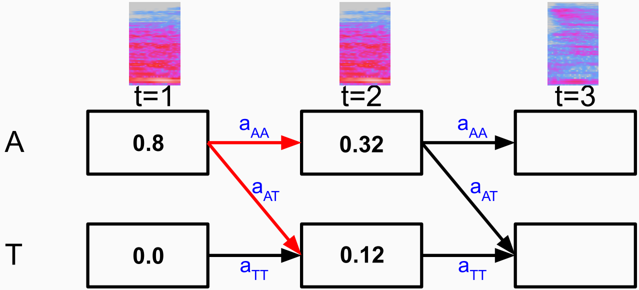


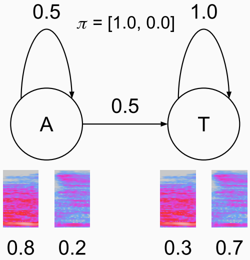


Answer

$Q =$

Viterbi algorithm, exercise 2 solution

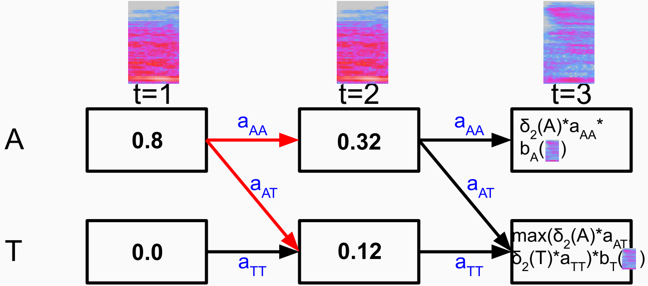


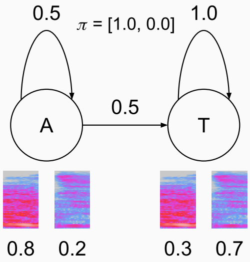


Answer

$Q =$

Viterbi algorithm, exercise 2 solution

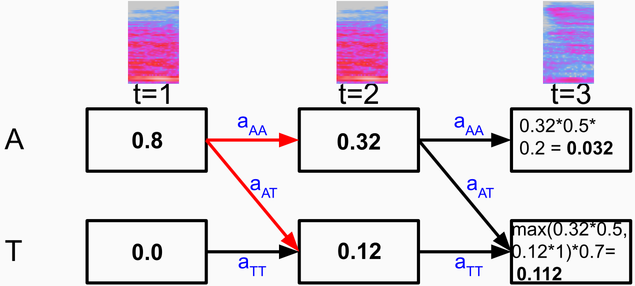


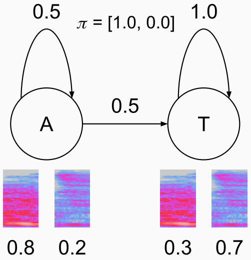


Answer

$Q =$

Viterbi algorithm, exercise 2 solution

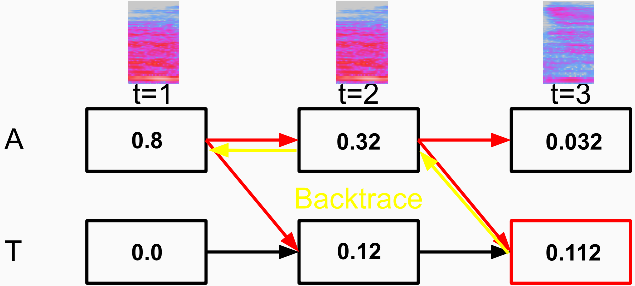


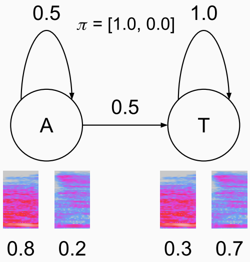


Answer

$Q =$

Viterbi algorithm, exercise 2 solution

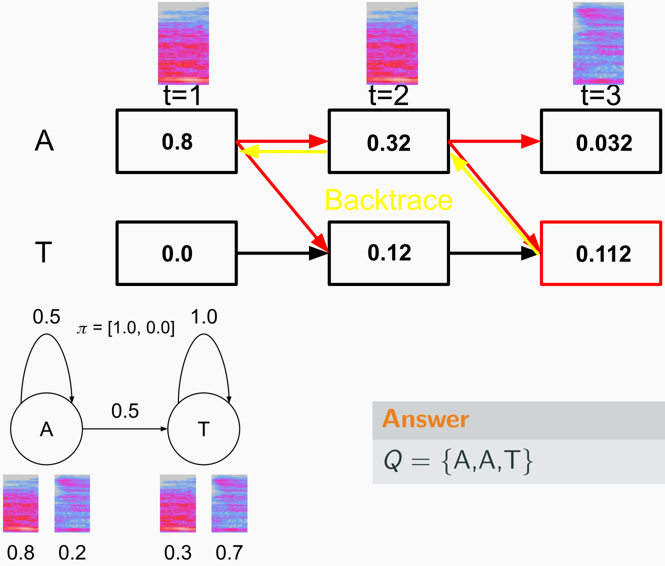




Answer

$Q =$

Viterbi algorithm, exercise 2 solution



Answer

$Q = \{\text{A,A,T}\}$

Practical tricks

The probabilities could get really small quickly, which leads to numerical stability issues$\rightarrow$Use **log** values!

There are many other regularization techniques

Leaky HMM

Allows transition from any state to any other state with a small probability (ϵ).

It's equivalent to stopping and restarting the HM on each frame.

Training

Baum-Welsh training

Forward-Backward algorithm

1. Initialize the model parameters (A, B)
2. Use the model and Forward (and Backward) algorithm to compute the probability matrix $P(q_t = i|A, B, O)$ for each sample
3. Update the model parameters using $P(q_t = i|A, B, O)$
4. Iterate from 2.

Viterbi training

- Similar to the Forward-Backward algorithm, substitutes $\sum$ operation with *max*
- Instead of summing probabilities over all HMM paths, only use the best path for each sample
- Technically uses "Hard alignment" vs the "soft alignment" in Forward-Backward
- Simpler, but converges likewise to the (local) optimum

Context-dependent HMM

Monophone HMMs ignore the context of the phoneme:
three = th + r + iy

Coarticulation complicate things! Solution: **context-dependent** HMM
Example **triphone** HMM: three = sil-th-r + th-r-iy+ r-iy-sil

CD also complicates things!How many possible triphones are in a system that has 40 monophones? **O(39*40*39)**

- How many models, states and Gaussians?
- Share models between some triphones?
- Share states or Gaussians between models?