Lecture Pattern Analysis

Part 20: HMMs Algorithm 3 and Remarks

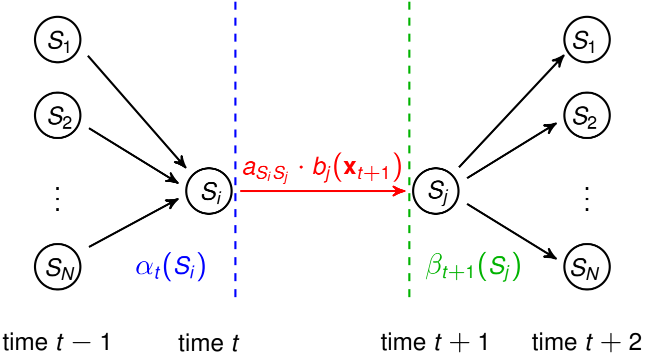
Christian Riess
IT Security Infrastructures Lab, Friedrich-Alexander-Universität Erlangen-Nürnberg
June 17, 2021

Overview

- The HMM training is an Expectation-Maximization algorithm
- For historical reasons, this EM is called Baum-Welch formulae
- The responsibilities are estimations for the current state or state transition, and their associated expectations
- Key idea:
  - Forward and backward algorithm can be used to marginalize out parameters
  - Use these algorithms to isolate every single parameter
- The maximization step updates the model parameters
- Key idea: counting relative frequencies converges to a local optimum
- As in the previous videos, $\mathbf{x}_t$ is a position in one input sequence $\mathbf{x}_1, \ldots, \mathbf{x}_T$
- However, the training is done over a dataset, so mentally add an outer loop around the calculations to accumulate the probabilities of all samples

Towards Responsibilities: Accessing Hidden State Transitions

- Recall that the state sequence is "hidden", and that we consider for HMM matching all state sequences
- For training, we have to update, e.g., an individual state transition $a_{S_i S_j}$
- This can be accessed by combining $\alpha_t(z_t = S_i)$ and $\beta_{t+1}(z_{t+1} = S_j)$:



Expectation Step: Calculation of the Responsibilities

- Probability for a state transition $S_i \rightarrow S_j$ at time t:

$$\xi_t(i,j) = p(z_t = S_i, z_{t+1} = S_j) \tag{1}$$

$$= \frac{\alpha_t(S_i) \cdot a_{S_i S_j} \cdot b_{t+1}(\mathbf{x}_{t+1})\beta_{t+1}(S_j)}{\sum_{z_t=S_1}^{S_N} \sum_{z_{t+1}=S_1}^{S_N} \alpha_t(z_t) \cdot a_{S_i S_j} \cdot b_{t+1}(\mathbf{x}_{t+1})\beta_{t+1}(z_{t+1})} \tag{2}$$

- Probability for being in state S_i at time t:

$$\gamma_t(i) = \sum_{j=1}^{N} \xi_t(i,j) \tag{3}$$

- Expected number of transitions $S_i \rightarrow S_j$ at all times: $\sum_{t=1}^{T} \xi_t(i,j)$
- Expected number of times S_i is visited: $\sum_{t=1}^{T} \gamma_t(i)$

Maximization Step

- Update of the starting probabilities:

$$\bar{\pi}_{S_i} = \gamma_1(i) = \text{expected \# times in } S_i \text{ in time } t \text{ over all training data} \tag{4}$$

- Update of the state transition probabilities:

$$\bar{a}_{S_i S_j} = \frac{\sum_{t=1}^{T} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)} = \frac{\text{exp. \# of transitions } S_i \rightarrow S_j}{\text{exp. \# of times in state } S_i} \tag{5}$$

- Update of the output probabilities (discrete alphabet):

$$\bar{b}_{S_i}(o_v) = \frac{\sum_{t=1}^{T} \gamma_t(i) \cdot \mathbb{I}(\mathbf{x}_t, o_v)}{\sum_{t=1}^{T} \gamma_t(S_i)} = \frac{\text{expected \# times in } S_i \text{ and observing } o_v}{\text{expected \# times in } S_i} \tag{6}$$

where $\mathbb{I}$ is the discrete indicator function (analogous to Dirac δ)

Remarks

- An HMM is called **ergodic** if every state can be reached from every other state
- In practice, **left-right HMMs** are much more commonly used. Left-right HMMs only allow forward edges and self-loops, i.e., $a_{s_i s_j} = 0$ if $i > j$
- It is not difficult to change the discrete alphabet to continuous observations
  - For example a GMM can model how well a state matches an observation (we use such a model in the hand-writing recognition exercise)
  - GMM training and HMM training work well together: both are fully probabilistic models, both are trained via EM
- Good parameter initializations before training can significantly improve results. For example, Rabiner (Sec. VI.F) proposes to pre-cluster speech segments for initial state assignments