CS 2750 Machine Learning
Lecture 24

Hidden Markov models

Milos Hauskrecht
milos@cs.pitt.edu
MIB 318

Hidden Markov models

- **Hidden Markov model** = doubly stochastic finite automaton

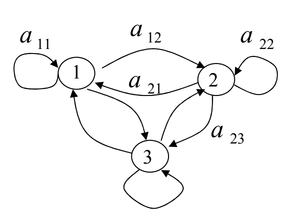
  States: $\mathbf{S} = \{s_i\}$

  Observations: $\mathbf{O} = \{o_k\}$

  Stochastic transitions

  $\mathbf{A} = \{a_{ij}\}$

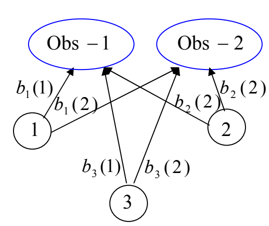
  $a_{ij} = P(s_{t+1} = j \mid s_t = i)$



  Observation model

  $\mathbf{B} = \{b_j(k)\}$

  $b_j(k) = P(o_t = k \mid s_t = j)$

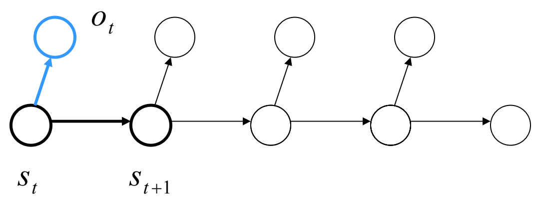


  Initial state distribution

  $\boldsymbol{\pi} = \{\pi_i\} \qquad \pi_i = P(s_1 = i)$

Hidden Markov models

- Graphical model representation (dynamic Bayesian belief network)



- **Three problems:**
  1. Given an observation sequence and the model compute the probability of the sequence
  2. Given the observation sequence compute the most likely (ML) hidden state sequence
  3. Learning of parameters of HMM (ML parameter estimate)

Inference in HMMs

- Let $O = \{O_1, O_2, ..O_T\}$ be a sequence of T observations
- Let $S = \{S_1, S_2, ..S_T\}$ be a sequence of internal states

$$P(O \mid S, \lambda) = b_{s_1}(O_1) b_{s_2}(O_2) \dots b_{s_T}(O_T)$$
$$P(S \mid \lambda) = \pi_{s_1} a_{s_1 s_2} a_{s_2 s_3} \dots a_{s_{T-1} s_T}$$

Then: $P(O \mid \lambda) = \sum_{S} P(O \mid S, \lambda) P(S \mid \lambda)$

Problem:

- The number of all possible sequences S is too large
- The exhaustive algorithm is exponential in T

HMM inference. Forward algorithm

Better solution: decompose the computation along the time

- Let $\alpha_t(i) = P(O_1, O_2, .. O_t, s_t = i \mid \lambda)$

Algorithm:

1. $\alpha_1(i) = \pi_i b_i(O_1)$ for all *i*
2. Repeat for *t =2,3, ... T-1* and all *j*

$$\alpha_{t+1}(j) = \left[\sum_i \alpha_t(i) a_{ij} \right]_i b_j(O_{t+1})$$

3. Then $P(O \mid \lambda) = \sum_i \alpha_T(i)$

This is called **forward algorithm**

- Implements a dynamic programming approach
- It is polynomial in the number of states and steps

HMM inference. Backward algorithm

- We can do the computation backward in time as well
- Let $\beta_t(i) = P(O_{t+1}, O_{t+2}, .. O_T \mid s_t = i, \lambda)$

Backward algorithm :

1. $\beta_T(i) = 1$ for all *i*
2. Repeat for *t =T-1,T-2, ... 1* and for all *i*

$$\beta_t(i) = \sum_j a_{ij} b_j(O_{t+1}(j)) \beta_{t+1}(j)$$

3. Then $P(O \mid \pi) = \sum_i \pi_i \beta_1(i)$

- The algorithm
  - is polynomial in the number of states and steps

Finding the most likely sequence

- **Goal:** compute the most likely sequence of states
- **Problem:** How to define the most likely sequence?
- **Solution 1: piece together most likely states**
- A probability of being in state *i* at time *t,* given the complete observation sequence

$$\gamma_t(i) = P(s_t = i \mid O, \lambda) = \frac{P(s_t = i, O \mid \lambda)}{P(O \mid \lambda)}$$

$$\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{P(O \mid \lambda)}$$

$$\alpha_t(i) = P(O_1, O_2, ..O_t, s_t = i \mid \lambda)$$

$$\beta_t(i) = P(O_{t+1}, O_{t+2}, ..O_T \mid s_t = i, \lambda)$$

$$s_t^* = \arg\max_i \gamma_t(i)$$

- **Problem:** what if the transition between the two most likely states is 0?

Viterbi algorithm

- **Goal:** compute the most likely sequence of states
- **Solution 2:** Find the best "continuous" sequence of states
  - Optimize with regard to:

$$P(O, S \mid \lambda)$$

  - Can be solved by dynamic programming
    - Similar to the forward algorithm

- **Viterbi algorithm** used frequently in the speech recognition

Viterbi algorithm

- **Initialization: (for all i)** $\delta_1(i) = \pi_i b_i(O_1)$

  $\psi_1(i) = 0$

- **Recursion (for all j)** $\delta_t(j) = \max_{1 \le i \le N} [\delta_{t-1} a_{ij}] b_j(O_t)$

  $\psi_t(j) = \max_{1 \le i \le N} [\delta_{t-1} a_{ij}]$

- **Termination** $P^* = \max_{1 \le i \le N} [\delta_T]$

  $s_T^* = \arg\max_{1 \le i \le N} [\delta_T]$

- **Sequence Recovery (T-1, …, 1)**

  $s_t^* = \psi_{t+1}[s_{t+1}^*]$

HMM learning

- Learning with hidden variables (the same idea as used in EM)
  - **Baum-Welch algorithm** is a special case of EM
- Find the ML set of parameters , maximizing $P(O \mid \lambda)$
- **HMM re-estimation step** (one cycle of EM):

$$\gamma_t(i) = P(s_t = i \mid O, \lambda) \qquad \xi_t(i,j) = P(s_t = i, s_{t+1} = j \mid O, \lambda)$$

$$\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{P(O \mid \lambda)} \qquad \xi_t(i,j) = \frac{\alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(i)}{P(O \mid \lambda)}$$

$$\tilde{\pi}_i = \gamma_1(i) \qquad \tilde{a}_{ij} = \frac{\sum_{t=1}^{T} \xi_t(i,j)}{\sum_{t=1}^{T} \gamma_t(i)} \qquad \tilde{b}_j(k) = \frac{\sum_{t=1, O_t = k}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)}$$

HMMs in practice

- HMMs have been widely used in various contexts
- **Speech recognition** (single word recognition)
  - words correspond to sequences of observations
  - we estimate a HMM for each word
  - the output model is a mixture of Gaussians over spectral features
- **Bio-sequence analysis**
  - a single HMM model for each type of protein (sequence of amino acids)
  - gene identication (parsing the genome)
  - etc.