CHAPTER 15: Hidden Markov Models

ETHEM ALPAYDIN
© The MIT Press, 2010

alpaydin@boun.edu.tr
http://www.cmpe.boun.edu.tr/~ethem/i2ml2e
Introduction

- Modeling dependencies in input; no longer iid
- Sequences:
  - Temporal: In speech; phonemes in a word (dictionary), words in a sentence (syntax, semantics of the language).
  - Spatial: In a DNA sequence; base pairs

In handwriting, pen movements
Discrete Markov Process

- **N states:** $S_1, S_2, \ldots, S_N$ State at “time” $t$, $q_t = S_i$
- **First-order Markov**
  \[ P(q_{t+1} = S_j \mid q_t = S_i, q_{t-1} = S_k, \ldots) = P(q_{t+1} = S_j \mid q_t = S_i) \]
- **Transition probabilities**
  \[ a_{ij} \equiv P(q_{t+1} = S_j \mid q_t = S_i) \quad a_{ij} \geq 0 \text{ and } \sum_{j=1}^{N} a_{ij} = 1 \]
- **Initial probabilities**
  \[ \pi_i \equiv P(q_1 = S_i) \quad \sum_{j=1}^{N} \pi_i = 1 \]
Stochastic Automaton
Example: Balls and Urns

• Three urns each full of balls of one color
  \( S_1: \) red, \( S_2: \) blue, \( S_3: \) green

\[
\Pi = [0.5, 0.2, 0.3] \quad A = \begin{bmatrix}
0.4 & 0.3 & 0.3 \\
0.2 & 0.6 & 0.2 \\
0.1 & 0.1 & 0.8
\end{bmatrix}
\]

\[
O = \{S_1, S_1, S_3, S_3\}
\]

\[
P(O | A, \Pi) = P(S_1) \times P(S_1 | S_1) \times P(S_3 | S_1) \times P(S_3 | S_3)
\]

\[
= \pi_1 \times a_{11} \times a_{13} \times a_{33}
\]

\[
= 0.5 \times 0.4 \times 0.3 \times 0.8 = 0.048
\]
Balls and Urns: Learning

- Given K example sequences of length T

\[
\pi_i = \frac{\# \{ \text{sequences starting with } S_i \}}{\# \{ \text{sequences} \}} = \frac{\sum_k 1(q^k_1 = S_i)}{K}
\]

\[
\hat{a}_{ij} = \frac{\# \{ \text{transitions from } S_i \text{ to } S_j \}}{\# \{ \text{transitions from } S_i \}} = \frac{\sum_k \sum_{t=1}^{T-1} 1(q^k_t = S_i \text{ and } q^k_{t+1} = S_j)}{\sum_k \sum_{t=1}^{T-1} 1(q^k_t = S_i)}
\]
Hidden Markov Models

- States are not observable
- Discrete observations \( \{v_1, v_2, ..., v_M\} \) are recorded; a probabilistic function of the state
- Emission probabilities
  \[
b_j(m) = P(O_t = v_m \mid q_t = S_j)\]
- Example: In each urn, there are balls of different colors, but with different probabilities.
- For each observation sequence, there are multiple state sequences
HMM Unfolded in Time
Elements of an HMM

- $N$: Number of states
- $M$: Number of observation symbols
- $A = [a_{ij}]: N \times N$ state transition probability matrix
- $B = b_j(m): N \times M$ observation probability matrix
- $\Pi = [\pi_i]: N \times 1$ initial state probability vector

$\lambda = (A, B, \Pi)$, parameter set of HMM
Three Basic Problems of

1. **Evaluation:** Given $\lambda$, and $O$, calculate $P(O | \lambda)$

2. **State sequence:** Given $\lambda$, and $O$, find $Q^*$ such that

$$P(Q^* | O, \lambda) = \max_Q P(Q | O, \lambda)$$

3. **Learning:** Given $X=\{O^k\}_k$, find $\lambda^*$ such that

$$P(X | \lambda^*) = \max_{\lambda} P(X | \lambda)$$

(Rabiner, 1989)
Evaluation

- Forward variable:

\[ \alpha_t(i) = P(O_t \square O_t, q_t = S_i \mid \lambda) \]

Initialization:

\[ \alpha_1(i) = \pi_i b_i(O_1) \]

Recursion:

\[ \alpha_{t+1}(j) = \left[ \sum_{i=1}^{N} \alpha_t(i) a_{ij} \right] b_j(O_{t+1}) \]

\[ P(O \mid \lambda) = \sum_{i=1}^{N} \alpha_T(i) \]
• **Backward variable:**

\[
\beta_t(i) = P(O_{t+1} \leq O_T | q_t = S_i, \lambda)
\]

**Initialization:**

\[
\beta_T(i) = 1
\]

**Recursion:**

\[
\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)
\]
Finding the State Sequence

\[ \gamma_t(i) \equiv P(q_t = S_i | O, \lambda) = \frac{\alpha_t(i) \beta_t(i)}{\sum_{j=1}^{N} \alpha_t(j) \beta_t(j)} \]

Choose the state that has the highest probability, for each time step:

\[ q_t^* = \arg \max_i \gamma_t(i) \]

No!
Viterbi’s Algorithm

\[ \delta_t(i) = \max_{q_1q_2 \ldots q_{t-1}} p(q_1q_2 \ldots q_{t-1}, q_t = S_i, O_1 \ldots O_t | \lambda) \]

- **Initialization:**
  \[ \delta_1(i) = \pi_i b_i(O_1), \psi_1(i) = 0 \]

- **Recursion:**
  \[ \delta_t(j) = \max_i \delta_{t-1}(i)a_{ij}b_j(O_t), \psi_t(j) = \arg\max_i \delta_{t-1}(i)a_{ij} \]

- **Termination:**
  \[ p^* = \max_i \delta_T(i), q_T^* = \arg\max_i \delta_T(i) \]

- **Path backtracking:**
  \[ q_t^* = \psi_{t+1}(q_{t+1}^*), t = T-1, T-2, \ldots, 1 \]
Learning

\[ \xi_t(i, j) = P(q_t = S_i, q_{t+1} = S_j \mid O, \lambda) \]

\[ \xi_t(i, j) = \frac{\alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)}{\sum_k \sum_l \alpha_t(k) a_{kl} b_l(O_{t+1}) \beta_{t+1}(l)} \]

Baum-Welch algorithm (EM):

\[ z_i^t = \begin{cases} 1 & \text{if } q_t = S_i \\ 0 & \text{otherwise} \end{cases} \quad z_{ij}^t = \begin{cases} 1 & \text{if } q_t = S_i \text{ and } q_{t+1} = S_j \\ 0 & \text{otherwise} \end{cases} \]
Baum–Welch (EM)

E-step: $E[Z_i^t] = \gamma_t^i \quad E[Z_{ij}^t] = \xi_t^{i,j}$

M-step:

$$\hat{\pi}_i = \frac{\sum_{k=1}^K \gamma_1^k(i)}{K}$$

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

$$\hat{b}_j(m) = \frac{\sum_{k=1}^K \sum_{t=1}^{T_k-1} \gamma_t^k(j) 1(O_t^k = v_m)}{\sum_{k=1}^K \sum_{t=1}^{T_k-1} \gamma_t^k(i)}$$
Continuous Observations

- **Discrete:**

\[ P(O_t \mid q_t = S_j, \lambda) = \prod_{m=1}^{M} b_j(m)^{r_m^t} \quad r_m^t = \begin{cases} 1 & \text{if } O_t = v_m \\ 0 & \text{otherwise} \end{cases} \]

- **Gaussian mixture (Discretize using k-means):**

\[ P(O_t \mid q_t = S_j, \lambda) = \sum_{l=1}^{L} P(G_j) P(O_t \mid q_t = S_j, G_l, \lambda) \]

- **Continuous:**

\[ P(O_t \mid q_t = S_j, \lambda) \sim \mathcal{N}(\mu_j, \Sigma_j) \]

\[ P(O_t \mid q_t = S_j, \lambda) \sim \mathcal{N}(\mu_j, \sigma_j^2) \]

Use EM to learn parameters, e.g.,

\[ \hat{\mu}_j = \frac{\sum_t \gamma_t(j)O_t}{\sum_t \gamma_t(j)} \]
HMM with Input

- Input–dependent observations:
  \[ P(O_t | q_t = S_j, x^t, \lambda) \sim \mathcal{N}(q_j(x^t | \theta_j) \sigma_j^2) \]

- Input–dependent transitions (Meila and Jordan, 1996; Bengio and Frasconi, 1996):
  \[ P(q_{t+1} = S_j | q_t = S_i, x^t) \]

- Time–delay input:
  \[ x^t = f(O_{t-\tau}, \ldots, O_{t-1}) \]
Model Selection in HMM

- Left-to-right HMMs:

\[
A = \begin{bmatrix}
  a_{11} & a_{12} & a_{13} & 0 \\
  0 & a_{22} & a_{23} & a_{24} \\
  0 & 0 & a_{33} & a_{34} \\
  0 & 0 & 0 & a_{44}
\end{bmatrix}
\]

- In classification, for each \( C_i \), estimate \( P(O \mid \lambda_i) \) by a separate HMM and use Bayes’ rule

\[
P(\lambda_i \mid O) = \frac{P(O \mid \lambda_i)P(\lambda_i)}{\sum_j P(O \mid \lambda_j)P(\lambda_j)}
\]