Hidden Markov Model and its applications in Signal Processing

Kanru Hua
Univ. of Illinois at Urbana-Champaign
Apr. 27, 2016

What is Hidden Markov Model?

- A hidden Markov model (HMM) is a statistical Markov model in which the system being modeled is assumed to be a Markov process with unobserved (hidden) states.

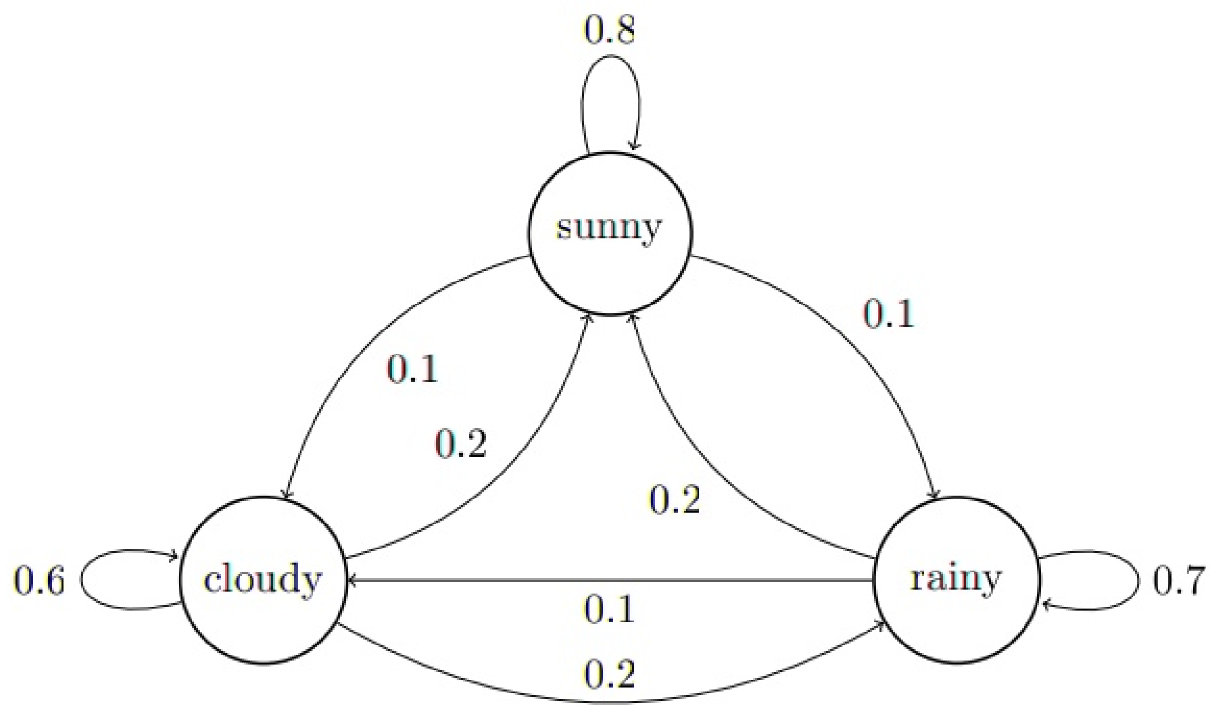
– Wikipedia

- **Typical applications:**
  - Speech Recognition/Synthesis
  - Part-of-speech Tagging
  - Handwriting Recognition
  - Automatic Audio/Video Transcription
  - Signal Denoising

Outline

- **Markov Chain**
- **Hidden Markov Model**
  - Definition
  - Algorithms – Inference and Training
  - Run HMM in Matlab
- **Extensions to HMM**
  - Continuous (and Multivariate) Output
  - Multiple Observation Sequences
- **Applications**
  - Speech Recognition
  - Filtering/Denoising

Markov Chain



- **Random walk example:**

  → Sunny → Sunny → Cloudy → Rainy → Rainy → Sunny → Sunny → Sunny → Rainy → Rainy → Cloudy → Sunny → ...

Markov Chain

- **Definition**

$$
\begin{aligned}
\text{parameter set:} \quad & \lambda = \{\pi, \mathbf{A}\} \\
\text{initial probability:} \quad & \pi_j = P(s_1 = j | \lambda) \\
\text{transition probability:} \quad & a_{ij} = P(s_t = j | s_{t-1} = i, \lambda)
\end{aligned}
$$

- **Markov Property**

$$P(s_t = j | s_1, s_2, ..., s_{t-2}, s_{t-1} = i, \lambda) = P(s_t = j | s_{t-1} = i, \lambda) = a_{ij}$$

Getting into current state only depends on the previous state but not any state before the previous state. (First-order Markov Chain)

Hidden Markov Model: an Example

On sunny days I usually go to ECEB by bike;

On cloudy days sometimes I walk.

On rainy days I either walk or take MTD, but I rarely bike.

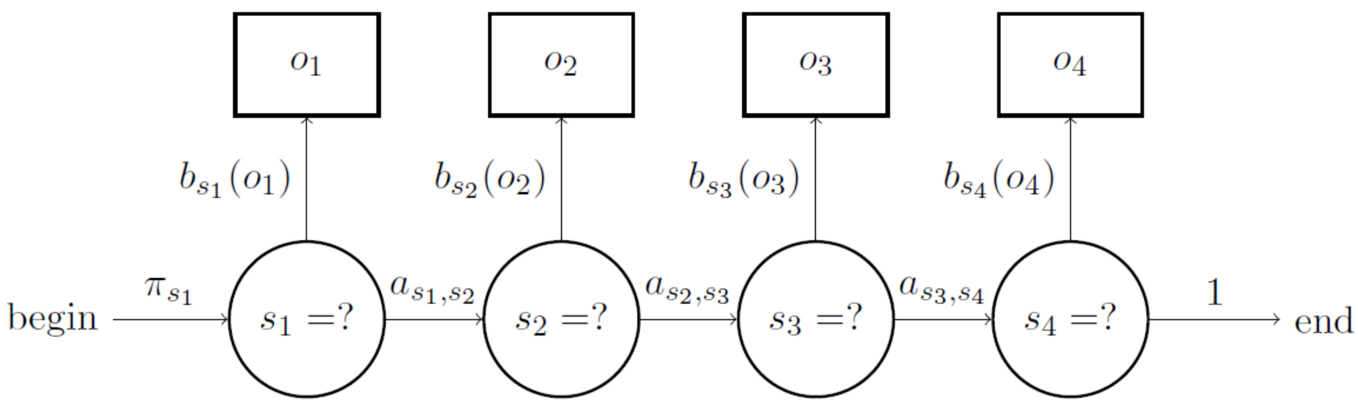
- My behavior can be modelled by a HMM and we can
  - Given weather data and my transportation record, train a HMM.
  - Given HMM & my transportation record, estimate the most probable weather sequence.
  - Given HMM & my transportation record, estimate the probability of having {sunny, cloudy, rainy} weather on each day.
  - Given HMM & transportation records of a few people at different locations, pick the one that is most likely to be my record (in Champaign).

Hidden Markov Model

- **Definition**

$$\begin{aligned} \text{parameter set:} \quad & \lambda = \{\pi, \mathbf{A}, \mathbf{B}\} \\ \text{initial probability:} \quad & \pi_j = P(s_1 = j|\lambda) \\ \text{transition probability:} \quad & a_{ij} = P(s_t = j|s_{t-1} = i, \lambda) \end{aligned}$$

$$\begin{aligned} \text{output probability (discrete case):} \quad & b_{jk} = P(o_t = k|s_t = j, \lambda) \\ \text{output probability density (continuous case):} \quad & b_j(o_t) = p(o_t|s_t = j, \lambda) \end{aligned}$$



Hidden Markov Model

- **What can we do with HMM?**
- **Generation**
  - Random walk
- **Inference**
  - Forward/Backward algorithm
  - Viterbi algorithm (most probable sequence)
  - Total probability and probability of a particular state
- **Parameter estimation (training)**
  - Initialization
  - Viterbi training algorithm
  - Baum-Welch algorithm

HMM Inference

- What we ultimately want to get:

$P(O\|\lambda)$	Total Probability
$P(s_t = j\|O, \lambda)$	State Occupancy Probability
$P(s_{t-1} = i, s_t = j\|O, \lambda)$	State Transition Probability
$\arg\max_{s_1,\ldots,s_t} P(o_1, \ldots, o_t, s_1, \ldots, s_t\|\lambda)$	Optimal Sequence

- Some intermediate values we'll be using:

$\alpha_t(i) = P(o_1, o_2, \ldots, o_t, s_t = i\|\lambda)$	Forward Probability
$\beta_t(i) = P(o_{t+1}, o_{t+2}, \ldots, o_t\|s_t = i, \lambda)$	Backward Probability

HMM Inference

- What we already know:

$$P(s_1 = i|\lambda) = \pi_i$$
$$P(s_t = j|s_{t-1} = i, \lambda) = a_{ij}$$
$$P(o_t|s_t = j, \lambda) = b_j(o_t)$$

- Directly following Markov property,

$$P(s_1, s_2, ..., s_t|\lambda) = \pi_{s_1} \prod_{\tau=2}^{t} a_{\tau-1,\tau}$$

$$P(o_1, o_2, ..., o_t|s_1, s_2, ..., s_t, \lambda) = \prod_{\tau=1}^{t} b_{s_\tau}(o_\tau)$$

$$P(o_1, o_2, ..., o_t, s_1, s_2, ..., s_t|\lambda) = \pi_{s_1} b_{s_1}(o_1) \prod_{\tau=2}^{t} a_{\tau-1,\tau} b_{s_\tau}(o_\tau)$$

Forward Algorithm

- Brute force way of getting forward probability:

$$
\begin{aligned}
&P(o_1, o_2, ..., o_t, s_t = j|\lambda) \\
&= \sum_{s_1,...,s_{t-1}} P(o_1, ..., o_t, s_1, ..., s_{t-1}, s_t = j|\lambda) \\
&= \sum_{\mathbf{s}} \pi_{s_1} b_{s_1}(o_1) \prod_{\tau=2}^{t} a_{s_{\tau-1},s_\tau} b_{s_\tau}(o_\tau)
\end{aligned}
$$

- Exponential time complexity!

Forward Algorithm

$$
\begin{aligned}
\alpha_t(j) &= P(o_1, o_2, ..., o_t, s_t = j|\lambda) \\
&= \sum_i P(o_1, ..., o_t, s_t = j, s_{t-1} = i|\lambda) \\
&= \sum_i P(o_1, ..., o_t|s_t = j, s_{t-1} = i, \lambda)P(s_t = j, s_{t-1} = i|\lambda) \\
&= \sum_i P(o_t|s_t = j, \lambda)P(o_1, ..., o_{t-1}|s_{t-1} = i, \lambda)P(s_t = j, s_{t-1} = i|\lambda) \\
&= P(o_t|s_t = j, \lambda) \sum_i P(o_1, ..., o_{t-1}, s_{t-1} = i|\lambda)P(s_t = j|s_{t-1} = i, \lambda) \\
&= b_j(o_t) \sum_i a_{ij}\alpha_{t-1}(i)
\end{aligned}
$$

Viterbi Algorithm

- Similarly, brute force won't work on $\arg\max_{s_1,\ldots,s_t} P(o_1, \ldots, o_t, s_1, \ldots, s_t|\lambda)$
- Need to define some intermediate values,

$$\begin{aligned}
\alpha_t^*(j) &= \max_{s_1,\ldots,s_{t-1}} P(o_1, \ldots, o_t, s_1, \ldots, s_{t-1}, s_t = j|\lambda) \\
&= \max_i \max_{s_1,\ldots,s_{t-2}} P(o_1, \ldots, o_t, s_1, \ldots, s_{t-2}, s_{t-1} = i, s_t = j|\lambda) \\
&= b_j(o_t) \max_i a_{ij}\alpha_{t-1}^*(i)
\end{aligned}$$

- Keep track of the paths we go through:

$$p_t^*(j) = \arg\max_i \, a_{ij}\alpha_{t-1}^*(i)$$

- Finally, backtrack from the last state to the first.

HMM Inference

- Total probability

$$P(O|\lambda) = \sum_j P(o_1, ..., o_T, s_t = j|\lambda) = \sum_j \alpha_T(j)$$

- State occupancy probability

$$\begin{aligned}\gamma_t(j) &= P(s_t = j|O, \lambda) \\ &= \frac{P(o_1, ..., o_T|s_t = j, \lambda)P(s_t = j|\lambda)}{P(O|\lambda)} \\ &= \frac{P(o_1, ..., o_t|s_t = j, \lambda)P(s_t = j|\lambda)P(o_{t+1}, ..., o_T|s_t = j, \lambda)}{P(O|\lambda)} \\ &= \frac{\alpha_t(j)\beta_t(j)}{P(O|\lambda)}\end{aligned}$$

Extension – Continuous Output

- Use a continuous density distribution for $b_j(o_t)$
- e.g. normal distribution

$$b_j(o_t = x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

- Each observation can also be a vector (using multivariate normal distribution)

$$b_j(o_t = x) = (2\pi)^{-\frac{k}{2}} |\Sigma|^{-\frac{1}{2}} e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)}$$

- To reduce number of parameters we often assume diagonal covariance, which significantly simplifies the whole thing.

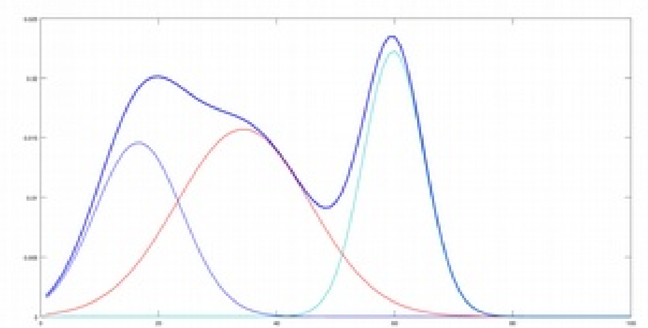
Extension – Continuous Output

- The inference algorithms are exactly same as the discrete case.
- Maximization step needs a little bit change:

$$\mu_j = \frac{\sum_{t=1}^{T} \gamma_t(j) o_t}{\sum_{t=1}^{T} \gamma_t(j)} \quad \Sigma_j = \frac{\sum_{t=1}^{T} \gamma_t(j) o_t o_t^T}{\sum_{t=1}^{T} \gamma_t(j)} - \mu_j \mu_j^T$$

- Gaussian Mixture Model as output density distribution

$$g_{jk}(o_t) = p(o_t | s_t = j, m_t = k, \lambda) = p(o_t | \boldsymbol{\mu}_{jk}, \boldsymbol{\Sigma}_{jk})$$
$$b_j(o_t) = p(o_t | s_t = j, \lambda) = \sum_k c_{jk} g_{jk}(o_t)$$



Extension – Multiple Observation Sequences

- Training data set often contains more than one sample (sequence).
- Use common sense, just add (average) bunch of things together!

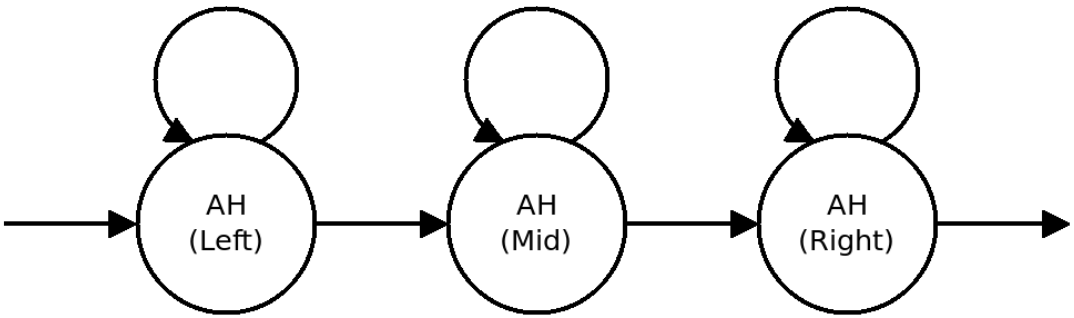
$$\pi_i = \frac{1}{L}\sum_{l=0}^{L} \gamma_1^l(i)$$

$$a_{ij} = \frac{\sum_{l=0}^{L}\sum_{t=2}^{T} \gamma_t^l(i,j)}{\sum_{l=0}^{L}\sum_{t=2}^{T} \gamma_{t-1}^l(i)} \qquad b_{ik} = \frac{\sum_{l=0}^{L}\sum_{t:o_t=k} \gamma_t^l(i)}{\sum_{l=0}^{L}\sum_{t=0}^{T} \gamma_t^l(i)}$$

- To prove this you can change the “s” under summation in EM algorithm and rederive the whole thing again.

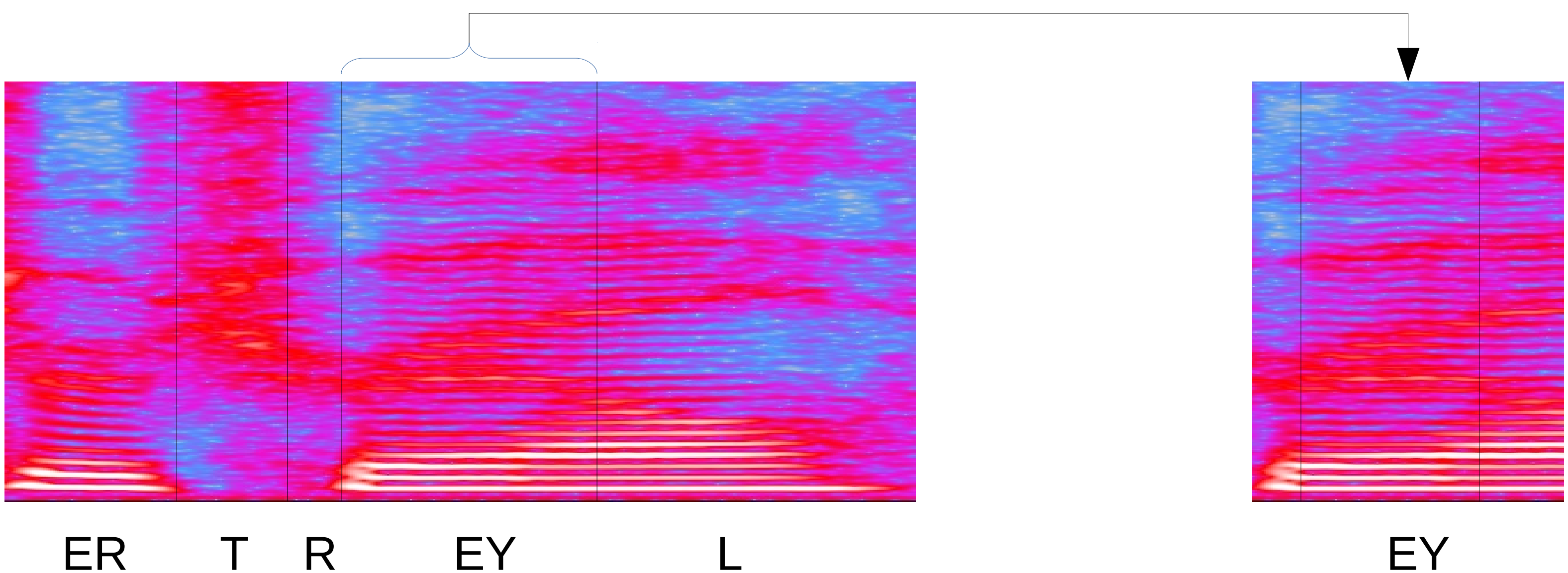
HMM for Speech Recognition

- **Triphone Models**



	Left	Mid	Right	Next
Left	0.7	0.3	0	0
Mid	0	0.9	0.1	0
Right	0	0	0.7	0.3

“LLLMMMMMMMRRN”

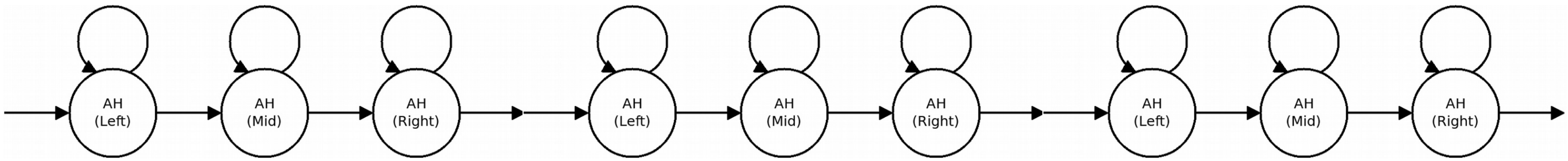


HMM for Speech Recognition

"Hello" → "HH AH L OW"

- We don't want the algorithm to consider impossible/confusing transitions (e.g. HH(R) → HH(L) or HH(R) → OW(L))

Then during inference, simply change the transition matrix for each training sample. Each state should only have *self transition* and *transition to the next state*. → Left-to-Right HMM



Forced alignment – given phoneme transcription and speech, calculate the starting/ending time of each phoneme (using Viterbi algorithm).

HMM for Speech Recognition

- **State Tying**

  What if a phoneme appears more than once in a sample?

  e.g. "Hello, world." → "HH AH L OW, W ER L D", where we can find L(R) → OW(L) and L(R) → D(L)

- Just create a copy of L(R, M, L) states, and name them L1(R, M, L)

  HH AH L OW, W ER L1 D

  L and L1 share the same output distribution. During maximization step, statistics on L and L1 are collected and grouped together. This trick is called state tying.

HMM for Speech Recognition

- **Context Dependency**

  So far we've only seen context-independent (CI) HMM – same L, M, R states are shared across the whole data set.

  However pronunciation could vary with respect to context.

  Full-context models: represent each occurrence of each phoneme by a unique set of L, M, R states.

- Context-dependent (CD) models: cluster full-context models into groups that share both similar context and similar output distribution, then merge & tie them together.

- We can use a decision tree for such clustering.