COMS 4721: Machine Learning for Data Science
Lecture 21, 4/13/2017

Prof. John Paisley

Department of Electrical Engineering
& Data Science Institute

Columbia University

HIDDEN MARKOV MODELS

Overview

Motivation

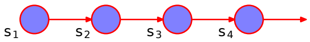
We have seen how Markov models can model sequential data.

- We assumed the observation was the sequence of states.
- Instead, each state may define a *distribution* on observations.

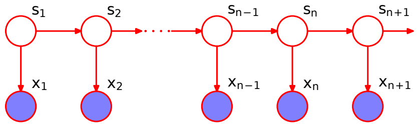
Hidden Markov model

A *hidden* Markov model treats a sequence of data slightly differently.

- Assume a hidden (i.e., unobserved, latent) sequence of states.
- An observation is drawn from the distribution associated with its state.



Markov model



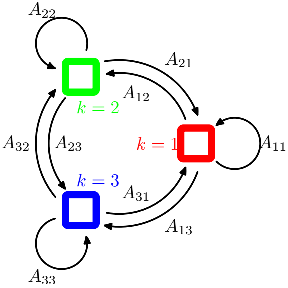
hidden Markov model

Markov models

Imagine we have three possible states in $\mathbb{R}^2$.
The data is a sequence of these positions.

Since there are only three unique positions, we can give an index in place of coordinates.

For example, the sequence $(1, 2, 1, 3, 2, \dots)$ would map to a sequence of 2-D vectors.



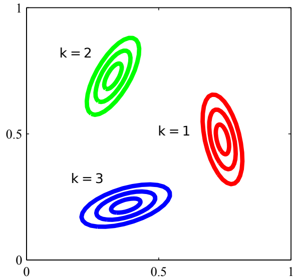
Using the notation of the figure, A is a 3×3 *transition matrix*. A_{ij} is the probability of transitioning from state i to state j.

Hidden Markov models

Now imagine the same three states, but each time the coordinates are randomly permuted.

The state sequence is still a set of indexes, e.g., $(1, 2, 1, 3, 2, \dots)$ of positions in $\mathbb{R}^2$.

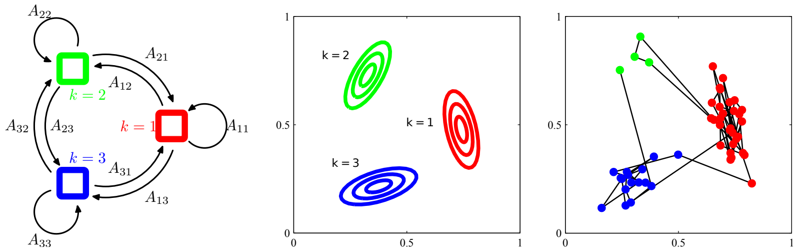
However, if μ_1 is the position of state #1, then we observe $x_i = \mu_1 + \epsilon_i$ if $s_i = 1$.



Exactly as before, we have a state transition matrix A (in this case 3×3).

However, the observed data is a sequence $(x_1, x_2, x_3, \dots)$ where each $x \in \mathbb{R}^2$ is a random perturbation of the state it's assigned to $\{\mu_1, \mu_2, \mu_3\}$.

Markov to Hidden Markov models



A continuous hidden Markov model

This HMM is *continuous* because each $x \in \mathbb{R}^2$ in the sequence $(x_1, \ldots, x_T)$.

(left) A Markov state transition distribution for an unobserved sequence

(middle) The state-dependent distributions used to generate observations

(right) The data sequence. Colors indicate the distribution (state) used.

Hidden Markov Models

Definition

A *hidden Markov model (HMM)* consists of:

- An $S \times S$ Markov transition matrix A for transitioning between S states.
- An initial state distribution π for selecting the first state.
- A state-dependent *emission distribution*, $\text{Prob}(x_i|s_i = k) = p(x_i|\theta_{s_i})$.

The model generates a sequence $(x_1, x_2, x_3 \dots)$ by:

1. Sampling the first state $s_1 \sim \text{Discrete}(\pi)$ and $x_1 \sim p(x|\theta_{s_1})$.
2. Sampling the Markov chain of states, $s_i|\{s_{i-1} = k\} \sim \text{Discrete}(A_{k,:})$, followed by the observation $x_i|s_i \sim p(x|\theta_{s_i})$.

Continuous HMM: $p(x|\theta_s)$ is a continuous distribution, often Gaussian.

Discrete HMM: $p(x|\theta_s)$ is a discrete distribution, θ_s a vector of probabilities.

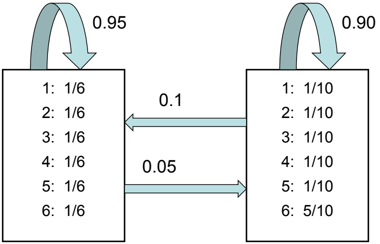
We focus on discrete case. Let B be a matrix, where $B_{k,:} = \theta_k$ (from above).

EXAMPLE: DISHONEST CASINO

Problem

Here is an example of a *discrete* hidden Markov model.

- Consider two dice, one is fair and one is unfair.
- At each roll, we either keep the current dice, or switch to the other one.
- The observation is the sequence of numbers rolled.



The transition matrix is

$$A = \begin{bmatrix} 0.95 & 0.05 \\ 0.10 & 0.90 \end{bmatrix}$$

The emission matrix is

$$B = \begin{bmatrix} \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} \\ \frac{1}{10} & \frac{1}{10} & \frac{1}{10} & \frac{1}{10} & \frac{1}{10} & \frac{1}{2} \end{bmatrix}$$

Let $\pi = [\frac{1}{2} \;\; \frac{1}{2}]$.

Some estimation problems

State estimation

- **Given:** An HMM $\{\pi, A, B\}$ and observation sequence $(x_1, \ldots, x_T)$
- **Estimate:** State probability for x_i using "forward-backward algorithm,"

$$p(s_i = k \,|\, x_1, \ldots, x_T, \pi, A, B).$$

State sequence

- **Given:** An HMM $\{\pi, A, B\}$ and observation sequence $(x_1, \ldots, x_T)$
- **Estimate:** Most probable state sequence using the "Viterbi algorithm,"

$$s_1, \ldots, s_T = \arg\max_{s} \; p(s_1, \ldots, s_T \,|\, x_1, \ldots, x_T, \pi, A, B).$$

Learn an HMM

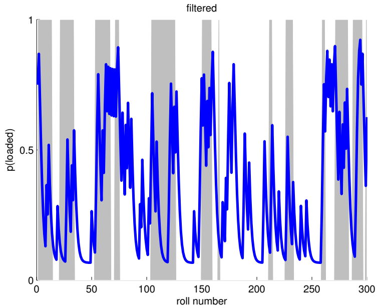
- **Given:** An observation sequence $(x_1, \ldots, x_T)$
- **Estimate:** HMM parameters π, A, B using maximum likelihood

$$\pi_{\text{ML}}, A_{\text{ML}}, B_{\text{ML}} = \arg\max_{\pi, A, B} \; p(x_1, \ldots, x_T \,|\, \pi, A, B)$$

EXAMPLES

Before we look at the details, here are examples for the dishonest casino.

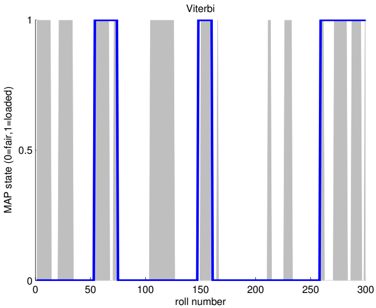
- Not shown is that π, A, B were learned first in order to calculate this.
- Notice that the right plot isn't just a rounding of the left plot.



State estimation result

Gray bars: Loaded dice used

Blue: Probability $p(s_i = \text{loaded} | x_{1:T}, \pi, A, B)$



State sequence result

Gray bars: Loaded dice used

Blue: Most probable state sequence

LEARNING THE HMM

Learning the HMM: The likelihood

We focus on the discrete HMM. To learn the HMM parameters, maximize

$$p(x|\pi, A, B) = \sum_{s_1=1}^{S} \cdots \sum_{s_T=1}^{S} p(x, s_1, \ldots, s_T \,|\, \pi, A, B)$$

$$= \sum_{s_1=1}^{S} \cdots \sum_{s_T=1}^{S} \prod_{i=1}^{T} p(x_i \,|\, s_i, B)\, p(s_i \,|\, s_{i-1}, \pi, A)$$

- $p(x_i \,|\, s_i, B) = B_{s_i, x_i} \leftarrow s_i$ indexes the distribution, x_i is the observation
- $p(s_i \,|\, s_{i-1}, \pi, A) = A_{s_{i-1}, s_i}$ (or π_{s_1}) $\leftarrow$ since $s_1, \ldots, s_T$ is a Markov chain

Learning the HMM: The log likelihood

- Maximizing $p(x|\pi, A, B)$ is hard since the objective has log-sum form

$$\ln p(x|\pi, A, B) = \ln \sum_{s_1=1}^{S} \cdots \sum_{s_T=1}^{S} \prod_{i=1}^{T} p(x_i \mid s_i, B)\, p(s_i \mid s_{i-1}, \pi, A)$$

- However, if we had or learned s it would be easy (remove the sums).

- In addition, we can calculate $p(s \mid x, \pi, A, B)$, though it's much more complicated than in previous models.

- Therefore, we can use the EM algorithm! The following is high-level.

LEARNING THE HMM: THE LOG LIKELIHOOD

E-step: Using $q(s) = p(s \mid x, \pi, A, B)$, calculate

$$\mathcal{L}(x, \pi, A, B) = \mathbb{E}_q \left[\ln p(x, s \mid \pi, A, B)\right].$$

M-Step: Maximize $\mathcal{L}$ with respect to π, A, B.

This part is tricky since we need to take the expectation using $q(s)$ of

$$\begin{aligned}
\ln p(x, s \mid \pi, A, B) \quad &= \quad \sum_{i=1}^{T} \sum_{k=1}^{S} \underbrace{\mathbb{1}(s_i = k) \ln B_{k,x_i}}_{\text{observations}} + \sum_{k=1}^{S} \underbrace{\mathbb{1}(s_1 = k) \ln \pi_k}_{\text{initial state}} \\
&\quad + \sum_{i=2}^{T} \sum_{j=1}^{S} \sum_{k=1}^{S} \underbrace{\mathbb{1}(s_{i-1} = j, s_i = k) \ln A_{j,k}}_{\text{Markov chain}}
\end{aligned}$$

The following is an overview to help you better navigate the books/tutorials.[1]

[1] See the classic tutorial: Rabiner, L.R. (1989). "A tutorial on hidden Markov models and selected applications in speech recognition." *Proceedings of the IEEE* **77**(2), 257–285.

Learning the HMM with EM

E-Step

Let's define the following conditional posterior quantities:

$$\begin{aligned}
\gamma_i(k) &= \text{the posterior probability that } s_i = k \\
\xi_i(j,k) &= \text{the posterior probability that } s_{i-1} = j \text{ and } s_i = k
\end{aligned}$$

Therefore, γ_i is a vector and ξ_i is a matrix, both varying over i.

We can calculate both of these using the "forward-backward" algorithm.
(We won't cover it in this class, but Rabiner's tutorial is good.)

Given these values the E-step is:

$$\mathcal{L} = \sum_{k=1}^{S} \gamma_1(k) \ln \pi_k + \sum_{i=2}^{T} \sum_{j=1}^{S} \sum_{k=1}^{S} \xi_i(j,k) \ln A_{j,k} + \sum_{i=1}^{T} \sum_{k=1}^{S} \gamma_i(k) \ln B_{k,x_i}$$

This gives us everything we need to update π, A, B.

M-Step

The updates for the HMM parameters are:

$$\pi_k = \frac{\gamma_1(k)}{\sum_j \gamma_1(j)}, \quad A_{j,k} = \frac{\sum_{i=2}^T \xi_i(j,k)}{\sum_{i=2}^T \sum_{l=1}^S \xi_i(j,l)}, \quad B_{k,v} = \frac{\sum_{i=1}^T \gamma_i(k) \mathbb{1}\{x_i = v\}}{\sum_{i=1}^T \gamma_i(k)}$$

The updates can be understood as follows:

- $A_{j,k}$ is the expected fraction of transitions $j \to k$ when we start at j
  - Numerator: *Expected* count of transitions $j \to k$
  - Denominator: *Expected* total number of transitions from j
- $B_{k,v}$ is the expected fraction of data coming from state k and equal to v
  - Numerator: *Expected* number of observations $= v$ from state k
  - Denominator: *Expected* total number of observations from state k
- π has interpretation similar to A

M-Step: N sequences

Usually we'll have multiple sequences that are modeled by an HMM. In this case, the updates for the HMM parameters with N sequences are:

$$\pi_k = \frac{\sum_{n=1}^{N} \gamma_1^n(k)}{\sum_{n=1}^{N} \sum_j \gamma_1^n(j)}, \quad A_{j,k} = \frac{\sum_{n=1}^{N} \sum_{i=2}^{T_n} \xi_i^n(j,k)}{\sum_{n=1}^{N} \sum_{i=2}^{T_n} \sum_{l=1}^{S} \xi_i^n(j,l)},$$

$$B_{k,v} = \frac{\sum_{n=1}^{N} \sum_{i=1}^{T_n} \gamma_i^n(k) \mathbb{1}\{x_i = v\}}{\sum_{n=1}^{N} \sum_{i=1}^{T_n} \gamma_i^n(k)}$$

The modifications are:

- Each sequence can be of different length, T_n
- Each sequence has its own set of γ and ξ values
- Using this we sum over the sequences, with the interpretation the same.

APPLICATION: SPEECH RECOGNITION

Application: Speech Recognition

Problem

Given speech in the form of an audio signal, determine the words spoken.

Method

- Words are broken down into small sound units (called *phonemes*). The states in the HMM are intended to represent phonemes.
- The incoming sound signal is transformed into a sequence of vectors (feature extraction). Each vector x_i is indexed by a time step i.
- The sequence $x_{1:T}$ of feature vectors is the data used to learn the HMM.

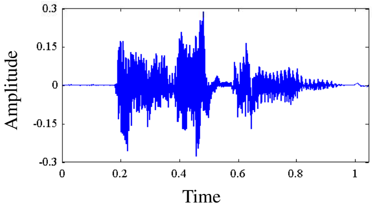
Phoneme

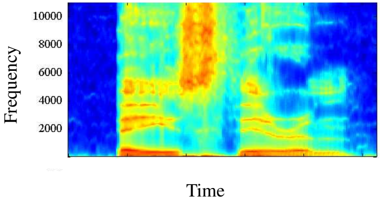
A phoneme is defined as the smallest unit of sound in a language that distinguishes between distinct meanings. English uses about 50 phonemes.

Example

Zero	Z IH R OW	Six	S IH K S
One	W AH N	Seven	S EH V AX N
Two	T UW	Eight	EY T
Three	TH R IY	Nine	N AY N
Four	F OW R	Oh	OW
Five	F AY V		

PREPROCESSING SPEECH

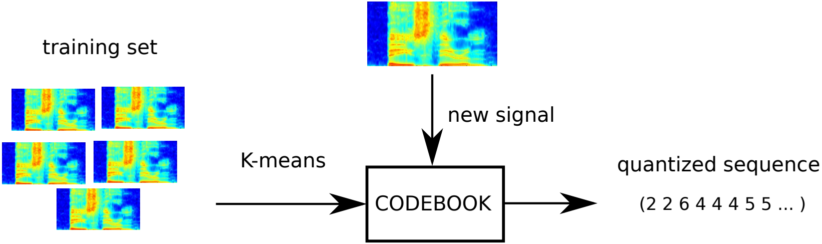




Feature extraction

- A speech signal is measured as amplitude over time.
- The signal is typically transformed into features by breaking down frequency content of the signal in a sliding time-window.
- (above) Each column is the frequency content of about 50 milliseconds (10,000+ dimensional). This can be further reduced to, e.g., 40 dims.

Data quantization



We could work directly with the extracted features and learn a Gaussian distribution for each state, i.e., a continuous HMM.

To transition to a discrete HMM, we can perform vector quantization using a codebook learned by K-means.

A SPEECH RECOGNITION MODEL

These models and problems can become more complex. For now, imagine a simple automated phone conversation using a question/answer format.

Training data: Quantized feature sequences of words, e.g., “yes,” “no”

Learn: An HMM for each word using all training sequences of that word

Predict: Let w index the word. Predict the word of a new sequence using

$$w_{new} = \arg\max_{w} \underbrace{p(x_{new} \mid \pi_w, A_w, B_w)}_{\text{requires forward-backward}} p(w)$$

Notice that this is a Bayes classifier!

- We’re learning a class-conditional discrete HMM.
- We could try something else, e.g., a GMM instead of an HMM.
- If the GMM predicts better, then use it instead. (But we anticipate that it won’t since the HMM models sequential information.)