ColumbiaX: Machine Learning
Lecture 21

Prof. John Paisley

Department of Electrical Engineering
& Data Science Institute

Columbia University

Hidden Markov Models

Overview

Motivation

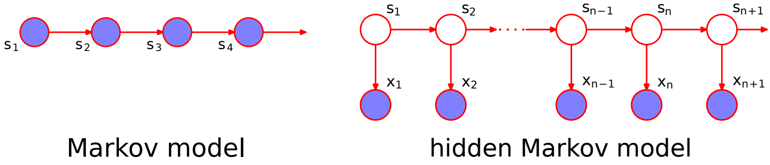
We have seen how Markov models can model sequential data.

- 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.

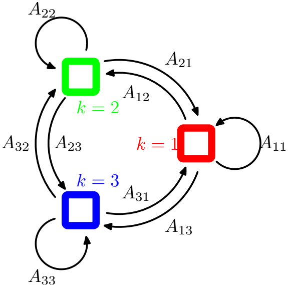


Markov models

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.



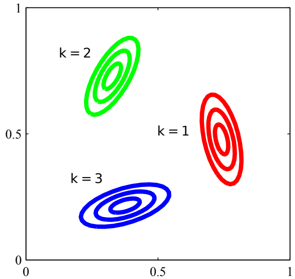
Using the notation of the figure, A is a 3×3 *transition matrix*. A_{ij} is the probability of transitioning from state i to state j.

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 $\mathbb{R}^2$.

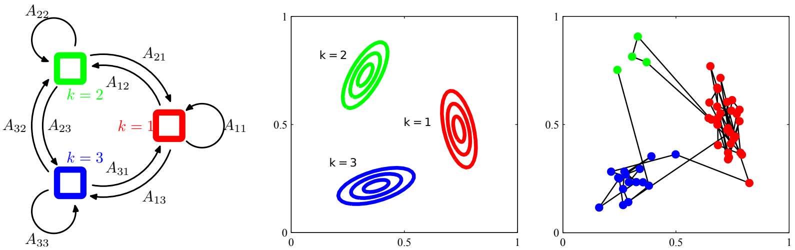
However, if μ_1 is the position of state $\#1$, then we observe $x_i = \mu_1 + \epsilon_i$ if $s_i = 1$.



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.

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.

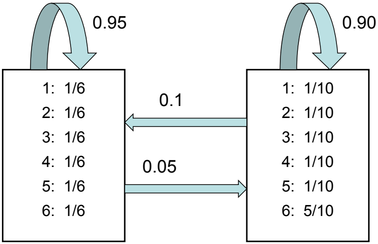
We focus on discrete case. Let B be a matrix, where $B_{s,:} = \theta_s$ (from above).

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 $(x_1, \dots, x_T)$
- **Estimate:** State probability for x_i using "forward-backward algorithm,"

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

State sequence

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

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

Learn an HMM

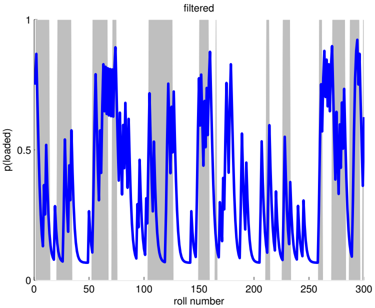
- **Given:** An observation sequence $(x_1, \dots, x_T)$
- **Estimate:** HMM parameters π, A, B using maximum likelihood

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

EXAMPLES

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

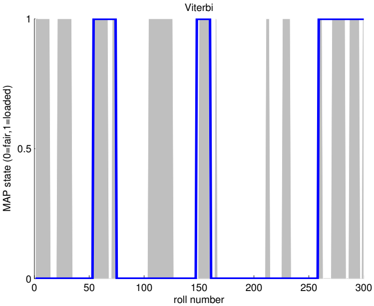
- Not shown is that π, A, B were learned first in order to calculate this.
- Notice that the right plot isn't just a rounding of the left plot.



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

$$
\begin{aligned}
p(\vec{x}|\pi, A, B) &= \sum_{s_1=1}^{S} \cdots \sum_{s_T=1}^{S} p(\vec{x}, s_1, \dots, s_T \,|\, \pi, A, B) \\
&= \sum_{s_1=1}^{S} \cdots \sum_{s_T=1}^{S} \prod_{i=1}^{T} p(x_i \,|\, s_i, B)\, p(s_i \,|\, s_{i-1}, \pi, A)
\end{aligned}
$$

- $p(x_i \,|\, s_i, B) = B_{s_i, x_i} \leftarrow s_i$ 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 $s_1, \dots, s_T$ is a Markov chain

Learning the HMM: The log likelihood

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

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

- However, if we had or learned $\vec{s}$ it would be easy (remove the sums).

- In addition, we can calculate $p(\vec{s} \mid \vec{x}, \pi, A, B)$, 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 $q(\vec{s}) = p(\vec{s} \,|\, \vec{x}, \pi, A, B)$, calculate

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

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

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

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

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.

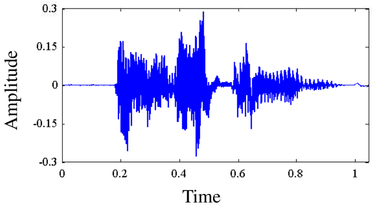
Phoneme

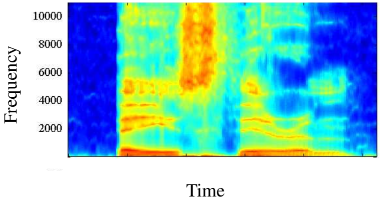
A phoneme is defined as the smallest unit of sound in a language that distinguishes between distinct meanings. English uses about 50 phonemes.

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

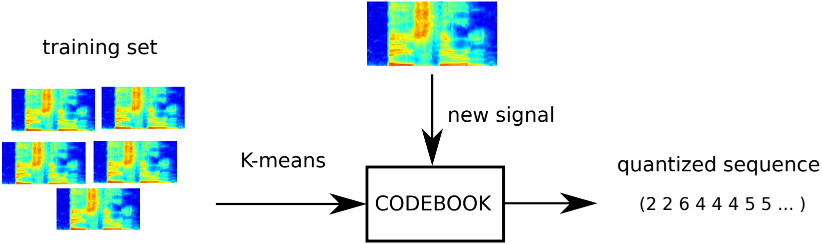




Feature extraction

- 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

$$w_{new} = \arg\max_{w}\ p(\vec{x}_{new} \,|\, \pi_w, A_w, B_w) \quad \leftarrow \text{requires forward-backward}$$

Notice that this is a Bayes classifier with a uniform prior on the word!

- 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.)

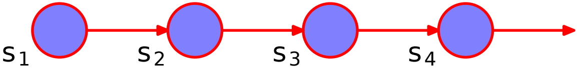
ColumbiaX: Machine Learning
Lecture 22

Prof. John Paisley

Department of Electrical Engineering
& Data Science Institute

Columbia University

Markov models



The sequence $(s_1, s_2, s_3, \dots)$ has the *Markov property*, if for all t

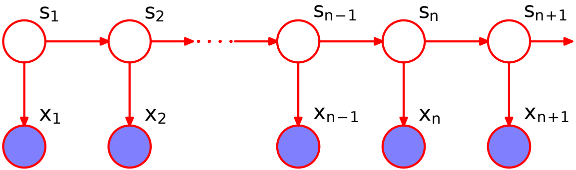
$$p(s_t | s_{t-1}, \dots, s_1) = p(s_t | s_{t-1}).$$

Our first encounter with Markov models assumed a finite state space, meaning we can define an indexing such that $s \in \{1, \dots, S\}$.

This allowed us to represent the transition probabilities in a matrix,

$$A_{ij} \quad \Leftrightarrow \quad p(s_t = j | s_{t-1} = i).$$

HIDDEN MARKOV MODELS



The hidden Markov model modified this by assuming the sequence of states was a *latent process* (i.e., unobserved).

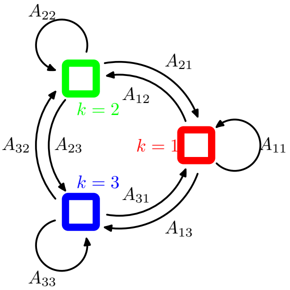
An observation x_t is associated with each s_t, where $x_t \mid s_t \sim p(x|\theta_{s_t})$.

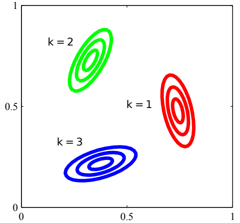
Like a mixture model, this allowed for a few distributions to generate the data. It adds an extra transition rule between distributions.

DISCRETE STATE SPACES

In both cases, the *state space* was discrete and relatively small in number.

- For the Markov chain, we gave an example where states correspond to positions in $\mathbb{R}^d$.
- A continuous hidden Markov model might perturb the latent state of the Markov chain.
  - For example, each s_i can be modified by continuous-valued noise, $x_i = s_i + \epsilon_i$.
  - But $s_{1:T}$ is still a *discrete* Markov chain.





Discrete vs continuous state spaces

Markov and hidden Markov models both assume a discrete state space.

For Markov models:

- The state could be a data point x_i (Markov Chain classifier)
- The state could be an object (object ranking)
- The state could be the destination of a link (internet search engines)

For hidden Markov models we can simplify complex data:

- Sequences of discrete data may come from a few discrete distributions.
- Sequences of continuous data may come from a few distributions.

What if we model the states as continuous too?

Continuous-state Markov model

Continuous Markov models extend the state space to a continuous domain. Instead of $s \in \{1, \ldots, S\}$, s can take any value in $\mathbb{R}^d$.

Again compare:

- Discrete-state Markov models: The states live in a discrete space.
- Continuous-state Markov models: The states live in a continuous space.

The simplest example is the process

$$s_t = s_{t-1} + \epsilon_t, \quad \epsilon_t \sim N(0, aI).$$

Each successive state is a perturbed version of the current state.

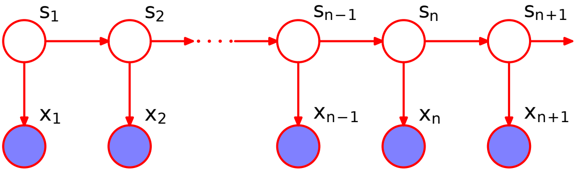
Linear Gaussian Markov model

The most basic continuous-state version of the hidden Markov model is called a *linear Gaussian Markov model* (also called the *Kalman filter*).

$$\underbrace{s_t = Cs_{t-1} + \epsilon_{t-1},}_{\text{latent process}} \qquad \underbrace{x_t = Ds_t + \varepsilon_t}_{\text{observed process}}$$

- $s_t \in \mathbb{R}^p$ is a continuous-state latent (unobserved) Markov process
- $x_t \in \mathbb{R}^d$ is a continuous-valued observation
- The process noise $\epsilon_t \sim N(0, Q)$
- The measurement noise $\varepsilon_t \sim N(0, V)$

EXAMPLE APPLICATIONS



Difference from HMM: s_t and x_t are *both* from continuous distributions.

The linear Gaussian Markov model (and its variants) has many applications.

- Tracking moving objects
- Automatic control systems
- Economics and finance (e.g., stock modeling)
- etc.

EXAMPLE: TRACKING

We get (very) noisy measurements of an object's position in time, $x_t \in \mathbb{R}^2$.

The time-varying state vector is $s = [\text{pos}_1\ \text{vel}_1\ \text{accel}_1\ \text{pos}_2\ \text{vel}_2\ \text{accel}_2]^T$.

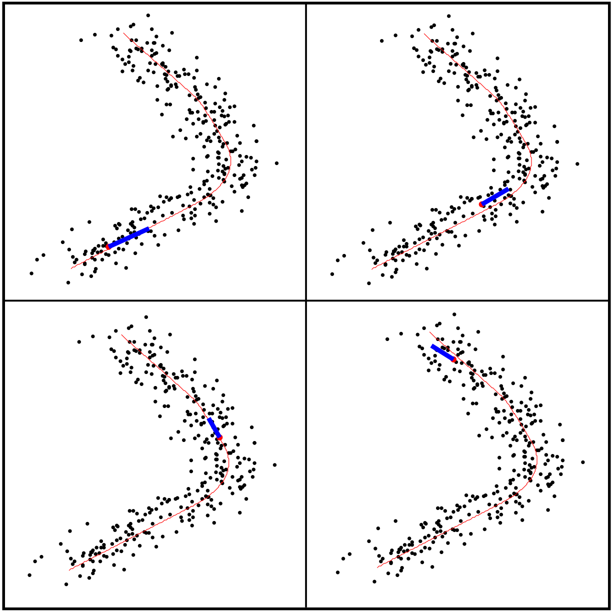
Motivated by the underlying physics, we model this as:

$$s_{t+1} = \underbrace{\begin{bmatrix} 1 & \Delta t & \frac{1}{2}(\Delta t)^2 & 0 & 0 & 0 \\ 0 & 1 & \Delta t & 0 & 0 & 0 \\ 0 & 0 & e^{-\alpha \Delta t} & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & \Delta t & \frac{1}{2}(\Delta t)^2 \\ 0 & 0 & 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 0 & 0 & e^{-\alpha \Delta t} \end{bmatrix}}_{\equiv \text{C}} s_t + \epsilon_t$$

$$x_{t+1} = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}}_{\equiv \text{D}} s_{t+1} + \varepsilon_{t+1}$$

Therefore, s_t not only approximates where the target is, but where it's going.

EXAMPLE: TRACKING



THE LEARNING PROBLEM

As with the hidden Markov model, we're given the sequence $(x_1, x_2, x_3, \dots)$, where each $x \in \mathbb{R}^d$. The goal is to learn state sequence $(s_1, s_2, s_3, \dots)$.

All distributions are Gaussian,

$$p(s_{t+1} = s|s_t) \; = \; N(Cs_t, Q), \qquad p(x_t = x|s_t) \; = \; N(Ds_t, V).$$

Notice that with the discrete HMM we wanted to learn π, A and B, where

- π is the initial state distribution
- A is the transition matrix among the discrete set of states
- B contains the state-dependent distributions on discrete-valued data

The situation here is very different.

THE LEARNING PROBLEM

No "B" to learn: In the linear Gaussian Markov model, each state is unique and so the distribution on x_t is different for each t.

No "A" to learn: In addition, each state transition is to a brand new state, so each s_t has its own unique probability distribution.

What we can learn are the two posterior distributions.

1. $p(s_t|x_1, \ldots, x_t)$: A distribution on the current state given the past.
2. $p(s_t|x_1, \ldots, x_T)$: A distribution on each latent state in the sequence

- #1: Kalman *filtering* problem. We'll focus on this one today.
- #2: Kalman *smoothing* problem. Requires extra step (not discussed).

The Kalman filter

Goal: Learn the sequence of distributions $p(s_t|x_1,\dots,x_t)$ given a sequence of data $(x_1,x_2,x_3,\dots)$ and the model

$$s_{t+1}\,|\,s_t \sim N(Cs_t, Q), \qquad x_t\,|\,s_t \sim N(Ds_t, V).$$

This is the (linear) Kalman filtering problem and is often used for tracking.

Setup: We can use Bayes rule to write

$$p(s_t|x_1,\dots,x_t) \;\propto\; p(x_t|s_t)\,p(s_t|x_1,\dots x_{t-1})$$

and represent the prior as a marginal distribution

$$p(s_t|x_1,\dots,x_{t-1}) = \int p(s_t|s_{t-1})\,p(s_{t-1}|x_1,\dots,x_{t-1})\,ds_{t-1}$$

THE KALMAN FILTER

We've decomposed the problem into parts that we do and don't know (yet)

$$p(s_t|x_1,\ldots,x_t) \;\propto\; \underbrace{p(x_t|s_t)}_{N(Ds_t,V)} \int \underbrace{p(s_t|s_{t-1})}_{N(Cs_{t-1},Q)} \, \underbrace{p(s_{t-1}|x_1,\ldots,x_{t-1})}_{?} \, ds_{t-1}$$

Observations and considerations:

1. The left is the posterior on s_t and the right has the posterior on s_{t-1}.
2. We want the integral to be in closed form and a known distribution.
3. We want the prior and likelihood terms to lead to a known posterior.
4. We want future calculations, e.g. for s_{t+1}, to be easy.

We will see how choosing the Gaussian distribution makes this all work.

THE KALMAN FILTER: STEP 1

Calculate the marginal for prior distribution

Hypothesize (temporarily) that the unknown distribution is Gaussian,

$$p(s_t|x_1,\ldots,x_t) \;\propto\; \underbrace{p(x_t|s_t)}_{N(Ds_t,V)} \int \underbrace{p(s_t|s_{t-1})}_{N(Cs_{t-1},Q)} \underbrace{p(s_{t-1}|x_1,\ldots,x_{t-1})}_{N(\mu,\Sigma)\ \textbf{by hypothesis}} \, ds_{t-1}$$

A property of the Gaussian is that marginals are still Gaussian,

$$\int N(s_t|Cs_{t-1},Q)N(s_{t-1}|\mu,\Sigma)ds_{t-1} = N(s_t|C\mu, Q + C\Sigma C^T).$$

We know C and Q (by design) and μ and Σ (by hypothesis).

THE KALMAN FILTER: STEP 2

Calculate the posterior

We plug in the marginal distribution for the prior and see that

$$p(s_t|x_1,\ldots,x_t) \;\propto\; N(x_t|Ds_t,V)\,N(s_t|C\mu,Q+C\Sigma C^T).$$

Though the parameters look complicated, the posterior is just a Gaussian

$$p(s_t|x_1,\ldots,x_t) = N(s_t|\mu',\Sigma')$$

$$\begin{aligned}
\Sigma' &= \left[(Q+C\Sigma C^T)^{-1} + D^T V^{-1} D\right]^{-1} \\
\mu' &= \Sigma'\left(D^T V^{-1} x_t + (Q+C\Sigma C^T)^{-1} C\mu\right)
\end{aligned}$$

We can plug the relevant values into these two equations.

Addressing the Gaussian assumption

By making the assumption of a Gaussian in the prior,

$$p(s_t|x_1,\ldots,x_t) \;\propto\; \underbrace{p(x_t|s_t)}_{N(x_t|Ds_t,V)} \int \underbrace{p(s_t|s_{t-1})}_{N(s_t|Cs_{t-1},Q)}\; \underbrace{p(s_{t-1}|x_1,\ldots,x_{t-1})}_{N(\mu,\Sigma)\ \textbf{by hypothesis}}\, ds_{t-1}$$

we found that the posterior is also Gaussian with a new mean and covariance.

- We therefore only need to define a Gaussian prior on the first state to keep things moving forward. For example,

  $$p(s_0) \sim N(0, I).$$

  Once this is done, all future calculations are in closed form.

KALMAN FILTER: ONE FINAL QUANTITY

Making predictions

We know how to update the sequence of state posterior distributions

$$p(s_t|x_1,\ldots,x_t).$$

What about predicting x_{t+1}?

$$\begin{aligned}
p(x_{t+1}|x_1,\ldots,x_t) &= \int p(x_{t+1}|s_{t+1})p(s_{t+1}|x_1,\ldots,x_t)ds_{t+1} \\
&= \int \underbrace{p(x_{t+1}|s_{t+1})}_{N(x_{t+1}|Ds_{t+1},V)} \int \underbrace{p(s_{t+1}|s_t)}_{N(s_{t+1}|Cs_t,Q)} \underbrace{p(s_t|x_1,\ldots,x_t)}_{N(s_t|\mu',\Sigma')} ds_t\, ds_{t+1}
\end{aligned}$$

Again, Gaussians are nice because these operations stay Gaussian.

This is a multivariate Gaussian that looks even more complicated than the previous one (omitted). Simply perform the previous integral twice.

ALGORITHM: KALMAN FILTERING

The Kalman filtering algorithm can be run in real time.

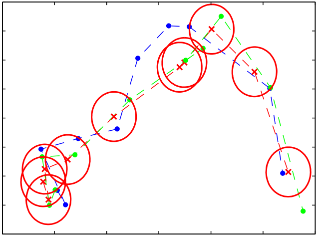
0. Set the initial state distribution $p(s_0) = N(0, I)$
1. Prior to observing each new $x_t \in \mathbb{R}^d$ predict

$$x_t \sim N(\mu_t^x, \Sigma_t^x) \qquad \text{(using previously discussed marginalization)}$$

2. After observing each new $x_t \in \mathbb{R}^d$ update

$$p(s_t | x_1, \dots, x_t) = N(\mu_t^s, \Sigma_t^s) \qquad \text{(using equations on previous slide)}$$

Example



Learning state trajectory

Green: True trajectory

Blue: Observed trajectory

Red: State distribution

Intuitions about what this is doing:

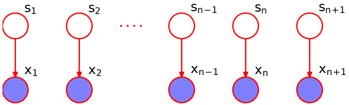
- In the prior distribution notice that we add Q to the covariance,

$$p(s_t|x_1,\ldots,x_{t-1}) = N(s_t|C\mu, Q + C\Sigma C^T).$$

  This allows the state s_t to "drift" away from s_{t-1}.

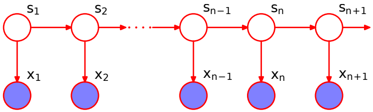
- In the posterior $p(s_t|x_1,\ldots,x_t)$, x_t "pulls" the distribution away.

Some final model comparisons



Gaussian mixture model

- $s_t \sim \text{Discrete}(\pi)$
- $x_t | s_t \sim N(\mu_{s_t}, \Sigma_{s_t})$

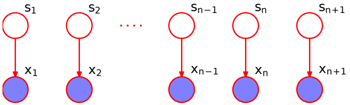


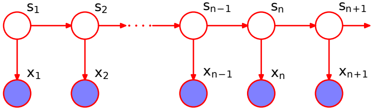
Continuous hidden Markov model

- $s_t | s_{t-1} \sim \text{Discrete}(A_{s_{t-1}})$
- $x_t | s_t \sim N(\mu_{s_t}, \Sigma_{s_t})$

We saw how the transition from GMM $\rightarrow$ HMM involves using a Markov chain to index the distribution on clusters.

Some final model comparisons





Probabilistic PCA

- $s_t \sim N(0, Q)$
- $x_t | s_t \sim N(Ds_t, V)$

Linear Gaussian Markov model

- $s_t | s_{t-1} \sim N(Cs_{t-1}, Q)$
- $x_t | s_t \sim N(Ds_t, V)$

There is a similar relationship between probabilistic PCA and the Kalman filter. (Probabilistic PCA also learns D, while the Kalman filter doesn't).

Extensions

There are a variety of extensions to this framework. The equations in the corresponding algorithms would all look familiar given our discussion.

Extended Kalman filter: *Nonlinear Kalman filters* use nonlinear function of the state, $h(s_t)$. The EKF approximates $h(s_t) \approx h(z) + \nabla h(z)(s_t - z)$

$$s_{t+1} \mid s_t \sim N(Ds_t, Q), \qquad x_t \mid s_t \sim N(h(s_t), V).$$

Continuous time: Sometimes the time between observations varies. Let Δ_t be the time between observation x_t and x_{t+1}, then model

$$s_{t+1} \mid s_t \sim N(s_t, \Delta_t Q), \qquad x_t \mid s_t \sim N(Ds_t, V).$$

Adding control: In dynamic models, we can add control to the state using a vector u_t whose values we choose (e.g., thrusters).

$$s_{t+1} \mid s_t \sim N(Cs_t + Gu_t, Q), \qquad x_t \mid s_t \sim N(Ds_t, V).$$