COMS 4721: Machine Learning for Data Science

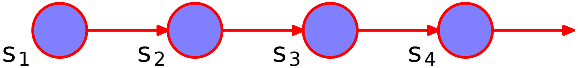
Lecture 22, 4/13/2021

Prof. John Paisley

Department of Electrical Engineering

Columbia University

Markov models



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

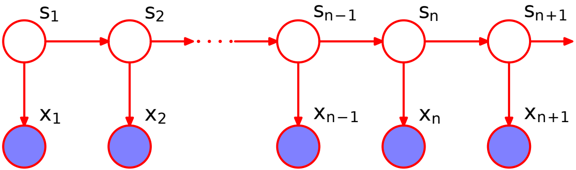
$$p(s_t | s_{t-1}, \dots, s_1) = p(s_t | s_{t-1}).$$

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

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).

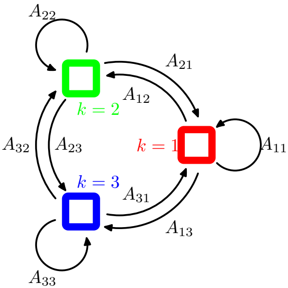
An observation x_t is associated with each s_t, where $x_t \mid s_t \sim p(x|\theta_{s_t})$.

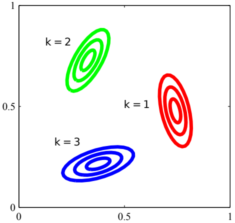
Like a mixture model, this allowed for a few distributions to generate the data. It adds an extra transition rule between distributions.

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 $s_t \in \{1, \ldots, S\}$, each s_t 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.

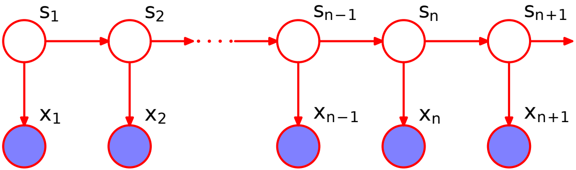
LINEAR GAUSSIAN MARKOV MODEL

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

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

- $s_t \in \mathbb{R}^p$ 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 $s = [\text{pos}_1 \ \text{vel}_1 \ \text{accel}_1 \ \text{pos}_2 \ \text{vel}_2 \ \text{accel}_2]^T$.

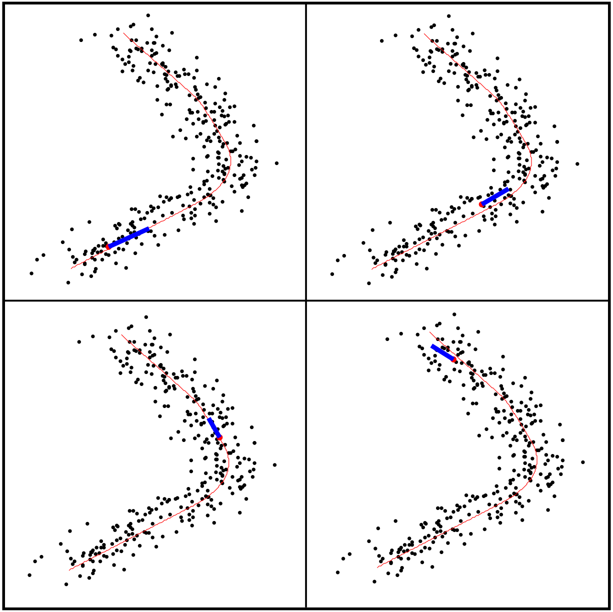
Motivated by the underlying physics, we model this as:

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

$$x_t = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}}_{\equiv \text{D}} s_t + \varepsilon_t$$

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 the state sequence $(s_1, s_2, s_3, \dots)$.

All distributions are Gaussian,

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

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. $p(s_t|x_1, \dots, x_t)$: A distribution on the current state given the past.
2. $p(s_t|x_1, \dots, x_T)$: 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

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

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,

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

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

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

We know C and Q by design, and μ_{t-1} and Σ_{t-1} by hypothesis.

THE KALMAN FILTER: STEP 2

Calculate the posterior

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

$$p(s_t|x_1,\ldots,x_t) \;\propto\; N(x_t|Ds_t,V)\,N(s_t|C\mu_{t-1},Q+C\Sigma_{t-1}C^T).$$

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

$$p(s_t|x_1,\ldots,x_t) = N(s_t|\mu_t,\Sigma_t)$$

$$\begin{aligned}\Sigma_t &= \left[(Q+C\Sigma_{t-1}C^T)^{-1}+D^TV^{-1}D\right]^{-1}\\ \mu_t &= \Sigma'\left(D^TV^{-1}x_t+(Q+C\Sigma_{t-1}C^T)^{-1}C\mu_{t-1}\right)\end{aligned}$$

We can plug the relevant values into these two equations.

Addressing the Gaussian assumption

By making the assumption of a Gaussian in the prior,

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

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

$$p(s_t|x_1,\dots,x_t) = N(s_t|\mu_t,\Sigma_t).$$

What about predicting x_{t+1}?

$$\begin{aligned}
p(x_{t+1}|x_1,\dots,x_t) &= \int p(x_{t+1}|s_{t+1})p(s_{t+1}|x_1,\dots,x_t)ds_{t+1} \\
&= \int \underbrace{p(x_{t+1}|s_{t+1})}_{N(x_{t+1}|Ds_{t+1},V)} \int \underbrace{p(s_{t+1}|s_t)}_{N(s_{t+1}|Cs_t,Q)} \underbrace{p(s_t|x_1,\dots,x_t)}_{N(s_t|\mu_t,\Sigma_t)} \, ds_t \, ds_{t+1}
\end{aligned}$$

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

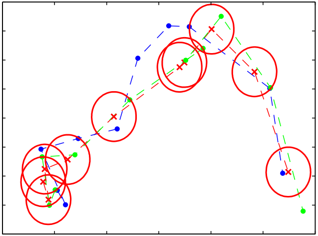
The Kalman filtering algorithm can be run in real time.

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

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



Learning state trajectory

Green: True trajectory

Blue: Observed trajectory

Red: State distribution

Intuitions about what this is doing:

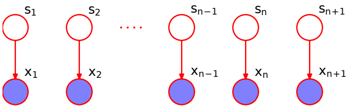
- In the prior distribution notice that we add Q to the covariance,

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

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

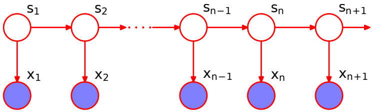
- In the posterior $p(s_t|x_1,\ldots,x_t)$, x_t "pulls" the distribution away.

Some final model comparisons



Gaussian mixture model

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

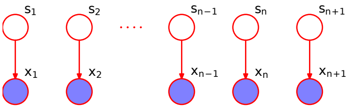


Continuous hidden Markov model

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

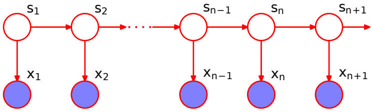
We saw how the transition from GMM $\rightarrow$ HMM involves using a Markov chain to index the distribution on clusters.

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 algorithms for the following have equations that would look familiar given our discussion.

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

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

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

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

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

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