# COMS 4721: Machine Learning for Data Science
## Lecture 22, 4/18/2017

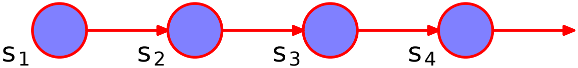
Prof. John Paisley

Department of Electrical Engineering
& Data Science Institute

Columbia University

. Sequential model where the state space is now continuous.

# Markov models




This sequence was observed, and modelled as

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

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

only conditionally dependent on previous time point.

Our first encounter with Markov models assumed a finite state space, [no. of states were finite and discrete.] meaning we can define an indexing such that $s \in \{1, \dots, S\}$.

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

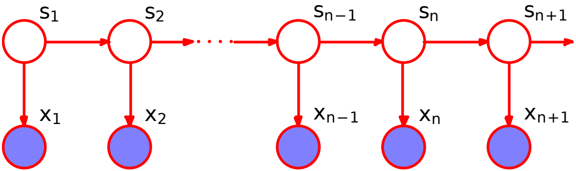
$$A_{ij} \quad \Leftrightarrow \quad p(s_t = j | s_{t-1} = i).$$

SxS matrix

Broadened the definition to define

# HIDDEN MARKOV MODELS

State sequence still followed a markov process, but the sequence itself was latent. What we observed state dependent observations from a state dependent observations.




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

observed at t

State dependent parameters.

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. (not present in the general mixture model framework.)

where distribution chosen at one time point was dependent on distribution chosen at previous time point.

# DISCRETE STATE SPACES

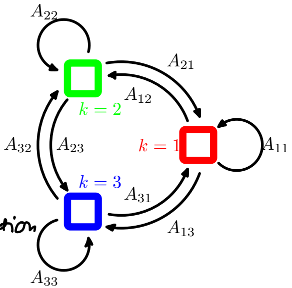
even data could be continuous.

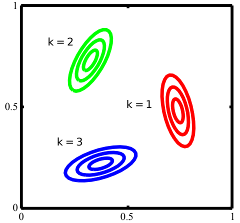
In both cases, the *state space* was discrete and relatively small in number. (finite )

- For the Markov chain, we gave an example where states correspond to positions in $\mathbb{R}^d$.

Generalize this, allow a particular state to be a random permutation where we had Gaussian noise.

- 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

What if we model the underlying state to be continuous.

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

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

noise is generated i.i.d from some Gaussian distribution.

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

Markov model because the latent state at time $t$ is only dependent on the latent space at time $t-1$.

# 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*).

Now, we have continuous functional form for the state process and the data process.

random noise unique to $t-1$

latent state at time $t$.

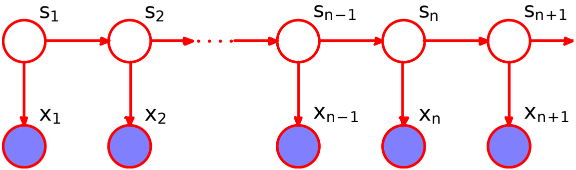
$$\underbrace{s_t = Cs_{t-1} + \epsilon_{t-1}}_{\text{latent process}}, \quad \underbrace{x_t = Ds_t + \varepsilon_t}_{\text{observed process}}$$

(unobserved 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 $\epsilon_t \sim N(0, Q)$ [ iid with mean 0, covariance Q Gaussian. ]
- The measurement noise $\varepsilon_t \sim N(0, V)$ [ " with different covariance ]

# EXAMPLE APPLICATIONS

In both case, we have exactly the same graphical representation for both the discrete state and continuous state markov model.




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.

The key difference is that the distribution on $S_2$ is a continuous probability distribution given $S_1$, whereas with hidden markov model, the distribution on $S_2$ was a discrete distribution on $S$ possible states given the state $S_1$.

So hidden Markov model had a discrete distribution on each of these states,
the continuous state Markov model has
a continuous distribution on each of these states.

# EXAMPLE: TRACKING

as a function of time

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

x, y coordinate

underlying state vector (unobserved state)

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

6-dimensional vector

Motivated by the underlying physics, we model this as:

2-dimensional position.

position in the first dimension of the object

times acceleration

state vector at time t+1

random perturbation.

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

position at prev. time point

amt. of × vel. time passed

observation

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

motion in that interval.

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

This is a simple Kalmen filter for modelling a moving object, try to use an inference algorithm to predict where the location is as well as predicting what the underlying state is but also velocity and acceleration as a function of time.

* What these 3 values are modelling is the position as a function of the previous position, the velocity in the $1^{st}$ dimension, the acceleration in the $1^{st}$ dimension.
  Give estimated position at time-step $\Delta t$ in the future.

** Second dimension of the state vector corresponds to velocity along the $1^{st}$ dimension of the observed space.

** damping effect → that says acceleration should be always either decreasing or increasing towards 0.
  → takes the old acceleration and multiplies by a number that's slightly less than 1 to dampen the acceleration.

- D matrix is picking out the position along the $1^{st}$ dimension and position along the second dimension of the state vector to say that the position is equal to whatever these 2 values are equal to plus noise.

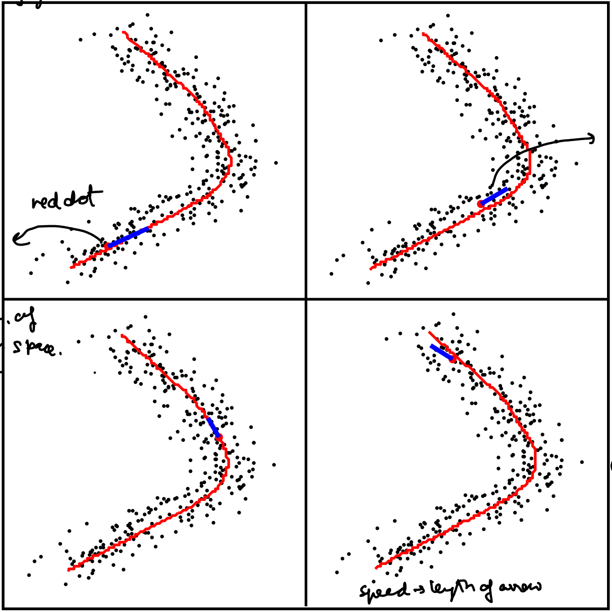
# EXAMPLE: TRACKING

[As a function of time we're learning with this tracking model, both the location & direction & speed of the movement of the object as function of time, based on only the black dot measurements.]

Can't tell the ordering of the measurements from the way they're plotted but you can tell they're noisy measurements.

Showing the state vector at a specific time point.

I$^{st}$ & II$^{nd}$ dimension of ↓ observation space. Position of object




blue line is showing the velocity vector.

Plotting the velocity in the I$^{st}$ dimension & velocity in the II$^{nd}$ dimension.

The blue line gives us the estimate of the direction and the speed in which the object is moving.

# 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 $(s_1, s_2, s_3, \dots)$.

Similar to HMM, learn what's going on with this hidden sequence. Unlike the HMM, the only thing we're learning is the underlying state sequence. We're not learning the emission distribution B, the transition matrix A, because now we have a continuous transition distribution

All distributions are Gaussian,

Another way of writing the generative process:

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

Distribution on state at time t+1 given state at time t. is equal to multivariate Gaussian with mean = $Cs_t$ & covariance = $Q$

observation at time t, given state at time t. We are assuming D/V are known or can be estimated.

Notice that with the discrete HMM we wanted to learn $\pi$, $A$ and $B$, where

- $\pi$ is the initial state distribution [had discrete set of states.]
- $A$ 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.

Here, we don't have any of those. All we want to learn is the underlying state vector as a function of time.

In previous problem we discussed, C & D were known from the physics of the problem. And Q and V were estimated somehow in advance.

# THE LEARNING PROBLEM

because the state in a continuous space with a continuous transition distribution, the distribution on each x is also going to be different. The way we contraint it by letting the distribution family be parameterized by us in this way. *

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

Because we don't have a discrete set of states **

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.

Learn the posterior distribution of the state at time T, given the sequence that we've observed up until time T. ***

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

all of the data including future data.

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

learning the continuously evolving distributions on the states in a real-time setting.

similar to backward algo. for HMM. This would be something where we have all data in advance and we want to do some sort of a post-processing.

However, because $S_t$ is continuous valued and unique, this distribution the parameters are going to be unique for every value of $X$.

$$p(x_t = x | S_t) = N(D S_t, V).$$

** $S$ is a continuous valued random vector with a continuous distribution, meaning that every state at every time point is going to be something brand new, a new continuous valued random vector. However, it's constrained by this distribution which is going to make the problem of learning the underlying state possible.

*** Distribution on the current state, given all of the data I have seen up until that time.

# THE KALMAN FILTER

Time evolving conditional posterior distribution of the state sequence

given the observed data up until that time for a given sequence $x_1$ through $x_n$.

**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

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

conditional posterior of my hidden state at time T, given all of data up until and including time T.

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

not including time Ts

and represent the prior as a marginal distribution

distribution

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

Distribution of state at time T given that data up until time T-1.

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

we simply add the state at t-1 and integrate it out to get a marginal.

* Underlying state at time t+1 given the underlying state at t is a multi-variate Gaussian

** Distribution on the observed values at time t given the underlying state at time t is a multi-variate Gaussian parameterize like this

*** But then we write the joint distribution as a conditional distribution on the state at time t given the state at t-1 times the posterior distribution on the state at time t-1

$p(s_t|s_{t-1})$

given the sequence up until time t-1.

Originally we had $x_1, \ldots x_{t-1}$ here. $p(s_t|s_{t-1})$, but if you tell me the state at time t-1, then the state at time t is conditionally independent of the sequence X up until the time t-1. So we could get of the sequence in the conditional here.

So we write this prior distribution $p(s_t|x_1 \ldots x_{t-1})$ as this marginal

$$\int \underbrace{p(s_t|s_{t-1})}_{\text{Transition distribution}} \underbrace{p(s_{t-1}|x_1, \ldots x_{t-1})}_{\text{posterior distribution for previous time point}} ds^{t-1}$$

?

So this is part of the story of base online learning was based in base rule
where the posterior at the previous time point
is used to form the prior at the next time point.

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

likelihood of observations at time t given state at time t

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

Posterior distribution on state at time t, given the observations up until and including time t

Transition distribution from time t-1 to time t

Posterior of the previous state at the last time given the observations at the last time point

Prior written as marginal

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.
   ↳ Meaning the posterior has a nice closed form solution and is in the same family as the prior.
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

Let's say that this conditional posterior at the last time point is a Gaussian with mean vector $\mu$ and covariance $\Sigma$ by hypothesis. And if this were true what will the solution to the posterior be.

## Calculate the marginal for prior distribution

Hypothesize (temporarily) that the unknown distribution is Gaussian,

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

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

Gaussian

Prior distribution on $s_{t-1}$

multiply & integrate out state at $t-1$

replace $s_{t-1}$ with the mean prior.

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

from this likelihood

Result is still a Gaussian

We know $C$ and $Q$ (by design) and $\mu$ and $\Sigma$ (by hypothesis).

(by model definition)

[We know $p(s_{t-1}|x_1 \ldots x_{t-1})$ $N(\mu,\Sigma)$ by hypothesis]

* Thus, this marginal likelihood to calculate $s_t$ is a Gaussian with a mean and a covariance, all of which we can calculate.

hinges on whether this likelihood times this prior is something that we can normalize and calculate the posterior.

## Calculate the posterior

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

likelihood of $x_t$ Prior on State at time t

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

calculated on previous slide.

covariance

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

So the posterior on $s_t$ is a Gaussian. So when we normalize this thing so that the function of $s_t$ integrates to 1. We find that we have a Gaussian with $\mu'$ & $\Sigma'$.

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

Prior covariance

function of everything we know.

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

observed data.

We can plug the relevant values into these two equations. & get the conditional posterior.

# 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.
equal to a function of things that we know

To turn that hypothesis into truth:

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

  Prior definition on the state at time 0, starting state.

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

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

Then by definition this 1st prior at $t=0$ is a multi-variate Gaussian and therefore everything follows through in order where the conditional posterior at $t=0$ is a multi-variate Gaussian because we have defined the prior at $t=0$ to be a multi-variate Gaussian.

 everything in Gaussian, can be calculated in closed form.

we have shown how to calculate the conditional posterior on state at time t given the observations up until time t. But what if we want to take and predict the observation at t+1 before we get to see it.

## Making predictions

We know how to update the sequence of state posterior distributions

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

$$\int p(x_{t+1}, s_{t+1} | x_{1 \ldots t}) \, ds_{t+1} \; *$$

data until time t

integrating out s at time t+1

What about predicting $x_{t+1}$?

what is the distribution on the observation at t+1, given all observations up to & including t

$$p(x_{t+1}|x_1,\ldots,x_t) = \int p(x_{t+1}|s_{t+1})p(s_{t+1}|x_1,\ldots,x_t)ds_{t+1} \quad **$$

Predictive distribution on the next observation.

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

⇒ This is how we take the predictive distribution and write it over distributions we know. by model definition; transition by def.; which is just the posterior distribution I have discussed how to calculate this. calculated using kalman filter algo. from previous slide

Again, Gaussians are nice because these operations stay Gaussian.

Integrating out $s_t$ & $s_{t+1}$ end up data at time t & up till time t+1.

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

of mean & covariance of things that we know.

* Joint distribution of the observation at time t+1 and the state time t+1.

→ Integrate out the state at t+1 to get this marginal distribution

$p(x_{t+1} \mid x_1 \ldots x_t)$

→ Write as a
- conditional distribution on observation at time t+1 given the state at time t+1. times the conditional distribution of the state at time t+1, given the data up until and including time t, but not t+1.

$p(x_{t+1} \mid s_{t+1})$ → Not conditioning on any other thing, because if you tell me what the state is time T+1. My observation at t+1 is independent of all other data.

→ * * write this as $\int p(s_{t+1}, s_t \mid x_{1-t}) \, ds_t$ Marginal of state at time t+1 and time t ) given the data up & until including T. But marginal out the state at time t.

factorize joint

***

# Algorithm: Kalman filtering

The Kalman filtering algorithm can be run in real time.

[arbitary Gaussian as long it is a Gaussian. Standard Gaussian here.]

0. Set the initial state distribution $p(s_0) = N(0, I)$

observation at $t$

1. Prior to observing each new $x_t \in \mathbb{R}^d$ predict

we form the predictive distribution. Prediction of what we are going to observe next.

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

at time $t$

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

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

update the conditional on the underlying state, given the data up until and including time $t$.

# EXAMPLE

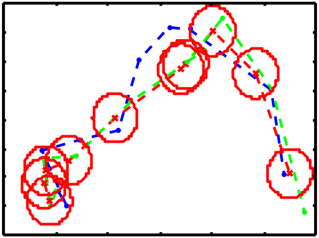
As a function of time, xs are the means of Gaussians, where we believe we are at timepoint and circles are the covariances.

## 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,

Prior distribution on the state at time t given the data up until and including time t-1.

adding some variance to the latent state. So at every timepoint we're adding some variance and allowing the state to drift away from what it was at time t-1. *

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

prior having not seen $x_t$

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

posterior having seen t

?

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

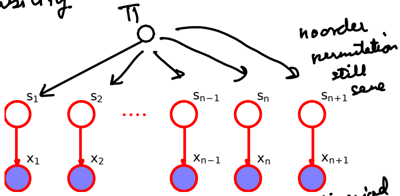
So that $x_t$ is pulled away by $s_t$ so it can model $x_t$ a little bit better. The prior constraints what $s_t$ can be, but adds a little bit of drift so it can move a little bit & be permuttted by $x_t$. **

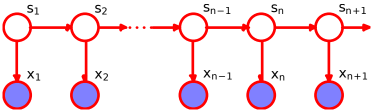
* This is what's making it a dynamic model, not something that's converging to a point estimate as $t$ increases. We're always adding the covariance to the state distribution. So we're always allowing the model to drift away from the previous state.

** And $x_t$ actually pulls the state in the direction that would have predicted it better.

# SOME FINAL MODEL COMPARISONS (Tie a few different models together.)

making simple modifications, get a fundamentally new modelling capability




discrete state markov model with continuous observed values

**Gaussian mixture model**

S different Gaussians. For time t, assign

- * $s_t \sim \text{Discrete}(\pi)$ data pt. to one of S Gaussians.
- $x_t | s_t \sim N(\mu_{s_t}, \Sigma_{s_t})$ This prob. distribution on S different Gaussians.

observation given cluster it was assigned to

pick out of state indexed.

**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.

* Pick a gaussian from a discrete distribution indexed by $S_t$

So I could also say that we have a vector pi here, and
then we simply pick a state based on this vector pi, this distribution pi.
And then given the state or given the cluster,
we generate the observation from the Gaussian picked out by that cluster.

** If we are at state $S_{t-1}$ at time $t-1$, then we transition to the state at time $S_t$, according to the distribution indexed by the row of our transition matrix (A) $S_{t-1}$.
Given that state, we generate our observation from a Gaussian, where the mean and covariance are picked out by the indexed state

Similarity:

1. So notice that given st in both of these models, we simply generate the data from a multivariate Gaussian with the meaning covariance picked out by the index S.

Difference:

1. However, for the Gaussian mixture model,
we're picking the clusters or picking the states ,
from a discrete distribution that's always using the same probability vector pi.
Whereas, for the continuous hidden Markov model,
which is a discrete state model, we're using also a discrete distribution, but
we're picking out the distribution according to a transition matrix.

Similarity

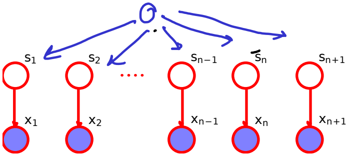
2. If there are capital S different states, we have S different probability distributions that we could possibly choose. And we choose the one indexed by the previous state in order to generate the next state, so they're very similar in that way.

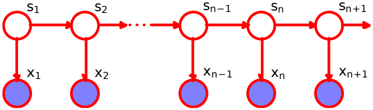
# SOME FINAL MODEL COMPARISONS

The previous example was comparison between 2 discrete state models. The GMM that had no sequential distribution and the continuous HMM which did have a sequential distribution on the discrete states.

Similar model structure as GMM

Take probabilistic PCA & build a linear model on top of it.




**Probabilistic PCA**

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

latent vector

Given the vector $s_t$, generate $t^{th}$ observation.

Latent state → a continuous random variable but there is no sequential information being modelled

**Linear Gaussian Markov model**

→ same covariance.

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

→ latent state at previous time point informs latent state at next time point. But given the latent state we have exactly the same distribution on observation as with prob. PCA.

but now it is a continuous valued vector instead of a discrete index.

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

Difference. prev. slide

So again, with discrete state models, Gaussian mixture model, and the continuous hidden Markov model, are essentially the same in the data generating distribution, but fundamentally different in the way that the distributions are picked.
One has a IID distribution on the way on the latent Gaussians.
And one has a sequential distribution on how these latent Gaussians are picked.

- Difference this slide

With a continuous state model, we have probabilistic PCA where again, the distributions are identical on the observations.
But on the latent states for probabilistic PCA, they're IID from a continuous Gaussian distribution, in which case, these aren't even referred to as states.
Whereas, with the linear Gaussian Markov model, we have a Markov sequence generating these latent states

# EXTENSIONS

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

approximation to this non-linear f(x) with linear f(x)

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

exactly the same state sequence as before

Generates observation using non-linear function of latent state

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

Brownian motion

calculating posterior of latent state no longer closed form.

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

time lapse from previous point. Q which is a constant matrix being scaled up/down based on how much time passed.

exactly same observation level distribution.

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

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

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

vector we control.

how we can control the latent space in order to control what we observe in the observation sequence $x_t$.