The sequence \((s_1, s_2, \ldots)\) has the \textit{Markov property}, if

\[
p(s_t|s_{t-1}, \ldots, s_1) = p(s_t|s_{t-1}).
\]

Our first encounter with Markov models assumed a finite state space, meaning we can index states such that \(s \in \{1, \ldots, S\}\).

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

\[
A_{ij} \iff p(s_t = j|s_{t-1} = i).
\]
The hidden Markov model extended this by assuming the sequence of states was a \textit{latent process} (i.e., it’s unobserved).

Associated with each $s_t$ was an observation $x_t$, where $x_t \sim p(x|\theta_{s_t})$.

The state of the $t$th observation, $s_t$, indexes the parameter used to generate the observation, $x_t$. This allowed for a few distributions to generate the data.
In both cases, the *state space* was discrete and relatively small in number.

- For the Markov chain, we can view the states as moving between 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 observation-specific noise, $x_i = s_i + \epsilon_i$.

  - But $s_{1:T}$ is still a *discrete* Markov chain.
Markov and hidden Markov models both assume a discrete state space.

For Markov models:
- The state could be a data point $x_i$ (MC classifier)
- The state could be an object (ranking)
- The state could be a link (internet search engines)

For hidden Markov models we can simplify complex data:
- Sequences of discrete data (e.g., codebook indexes) may be generated from a small set of discrete distributions, which groups similar codes.
- Sequences of continuous data may come from a few distributions.

What if we model the states as continuous too?
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:

- Continuous HMM: The observation $x_i \sim p(x|\theta_{s_i})$ is continuous-valued, but the state space of $s_i$ is discrete.
- 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.
The most basic continuous-state version of the hidden Markov model is called a *linear Gaussian Markov model* (among other names).

\[
\begin{align*}
  s_{t+1} &= C s_t + \epsilon_t, \\
  x_t &= D s_t + \varepsilon_t
\end{align*}
\]

- \( s_t \in \mathbb{R}^p \) is a continuous-state (hidden) Markov process
- \( x_t \in \mathbb{R}^d \) is a continuous-valued observation
- The noise \( \epsilon_t \sim N(0, aI) \) and \( \varepsilon_t \sim N(0, bI) \).

This is also referred to as the *Kalman filter*. 
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 objects such as faces or missiles.
- Automatic control systems
- Economics and finance (e.g., stock modeling)
- etc.
THE LEARNING PROBLEM

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

All distributions are Gaussian,

\[
p(s_{t+1} = s \mid s_t) = N(Cs_t, aI), \quad p(x_t = x \mid s_t) = N(Ds_t, bI).
\]

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

- \(\pi\) 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 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 transition is to a new state, so each $s_t$ has its own unique probability distribution.

What we can learn are the two posterior distributions.

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

Learning this second distribution is called the filtering problem.

- Parallels “forward” step of the forward-backward algorithm (HMM).
- This distribution is simpler to learn and we will focus on it today.
- The first distribution requires an additional “backward” step.
**The Kalman Filter**

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

\[
s_{t+1}|s_t \sim N(Cs_t, aI), \quad x_t|s_t \sim N(Ds_t, bI).
\]

This is often used for tracking and is called a *Kalman filter*.

**Setup:** We can use Bayes rule to write

\[
p(s_t|x_1, \ldots, x_t) \propto p(x_t|s_t) p(s_t|x_1, \ldots x_{t-1})
\]

and represent the prior as a marginal distribution

\[
p(s_t|x_1, \ldots, x_{t-1}) = \int p(s_t|s_{t-1}) p(s_{t-1}|x_1, \ldots, x_{t-1}) \, ds_{t-1}
\]
We’ve decomposed the problem into parts we do and don’t know (yet)

\[ p(s_t|x_1, \ldots, x_t) \propto p(x_t|s_t) \int p(s_t|s_{t-1}) p(s_{t-1}|x_1, \ldots, x_{t-1}) \, ds_{t-1} \]

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,

\[ p(s_t|x_1, \ldots, x_t) \propto p(x_t|s_t) \int_{N(Ds_t,bI)} p(s_t|s_{t-1}) p(s_{t-1}|x_1, \ldots, x_{t-1}) \, ds_{t-1} \]

\[ \int_{N(Cs_{t-1},aI)} N(s_t|C\mu, aI + C\Sigma C^T) ds_{t-1} = N(s_t|C\mu, aI + C\Sigma C^T) \]

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

We know \( C \) and \( a \) (by design) and \( \mu \) and \( \Sigma \) (by hypothesis).
Calculate the posterior

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

\[ p(s_t|x_1, \ldots, x_t) \propto N(x_t|D s_t, bI) N(s_t|C \mu, aI + C \Sigma C^T). \]

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

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

\[
\Sigma' = \left( (aI + C \Sigma C^T)^{-1} + D^T D / b \right)^{-1}
\]

\[
\mu' = \Sigma' \left( D^T x_t / b + (aI + C \Sigma C^T)^{-1} C \mu \right)
\]

We can plug the relevant values into these two equations.
By making the assumption of a Gaussian in the prior,

\[
p(s_t|x_1, \ldots, x_t) \propto p(x_t|s_t) \int p(s_t|s_{t-1}) \ p(s_{t-1}|x_1, \ldots, x_{t-1}) \ ds_{t-1}
\]

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

- Incrementing \( t \) by one, we see that the Gaussian posterior we’ve just calculated at time \( t \) fills the same role in the prior at time \( t + 1 \).

- We therefore only need to define a Gaussian prior on the first state to keep this ball rolling, e.g.,

\[
p(s_0) \sim N(0, I).
\]

Once this is done, all downstream equations 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 \( x_{t+1} \)?

\[
p(x_{t+1} | x_1, \ldots, x_t) = \int p(x_{t+1} | s_{t+1}) p(s_{t+1} | x_1, \ldots, x_t) ds_{t+1}
\]

\[
= \int p(x_{t+1} | s_{t+1}) \int p(s_{t+1} | s_t) p(s_t | x_1, \ldots, x_t) ds_t ds_{t+1}
\]

Again, Gaussians are nice because these operations stay Gaussian.

This is a multivariate Gaussian that looks even more complicated than the last one, but is just a function of things we know (omitted).
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

   $$x_t \sim N(\mu^x_t, \Sigma^x_t)$$

   (using previously discussed marginalization)

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

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

   (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 $aI$ to the covariance,

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

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

- In the posterior $p(s_t|x_1, \ldots, x_t)$, $x_t$ “drags” the distribution towards $x_t$.

- This dragging is aided by the drift $aI$, but also constrained by prior.
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.
There is a similar relationship between probabilistic PCA and the Kalman filter. Aside from the Markov chain difference, probabilistic PCA learns $W$. 

**Probabilistic PCA**

- $s_t \sim N(0, aI)$
- $x_t | s_t \sim N(Ws_t, bI)$

**Linear Gaussian Markov model**

- $s_t | s_{t-1} \sim N(Cs_{t-1}, aI)$
- $x_t | s_t \sim N(Ds_t, bI)$
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:** The dynamics can involve non-linear functions. The EKF uses a Taylor expansion approximation to learn \( s_t \) easily.

\[
\begin{align*}
  s_{t+1} \mid s_t & \sim f(s_t, \Theta), \\
  x_t \mid s_t & \sim g(s_t, \Phi).
\end{align*}
\]

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

\[
\begin{align*}
  s_{t+1} \mid s_t & \sim N(s_t, a\Delta_t I), \\
  x_t \mid s_t & \sim N(Ds_t, bI).
\end{align*}
\]

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

\[
\begin{align*}
  s_{t+1} \mid s_t & \sim N(Cs_t + Gu_t, a\Delta_t I), \\
  x_t \mid s_t & \sim N(Ds_t, bI).
\end{align*}
\]