Consider:

**Our recursive Bayes estimator from the homework.**

Let's write it as a graphical model.

\[
\begin{array}{c}
X \\

Y_1 \quad \cdots \quad Y_n
\end{array}
\]

They say all \( Y_i \) are i.i.d. measurements of unknown variable (state) \( X \).

Namely

\[
P(Y_1, Y_2, \ldots, Y_n | X) = P(Y_1 | X) P(Y_2 | X) \cdots P(Y_n | X)
\]

**Read off the joint distribution**

\[
P(X, Y_1, \ldots, Y_n) = P(X) P(Y_1 | X) \cdots P(Y_n | X) = P(X) P(Y_1, \ldots, Y_n | X)
\]

**Batch mode Bayes rule:**

\[
P(X, Y_1, \ldots, Y_n) = \frac{P(X, Y_1, \ldots, Y_n)}{\sum_X P(X, Y_1, \ldots, Y_n)} = \frac{P(X) P(Y_1, \ldots, Y_n | X)}{\sum_X P(X) P(Y_1, \ldots, Y_n | X)}
\]

Now consider recursive application of Bayes' rule based on \( P(X) P(Y_1 | X) \cdots P(Y_n | X) \)

**Prior likelihood**

\[
P(X) P(Y_1 | X) = \frac{P(X, Y_1)}{\sum_X P(X, Y_1)} = \frac{P(X, Y_1)}{P(Y_1)} = P(X | Y_1)
\]

**Prior likelihood**

\[
P(X, Y_1, Y_2 | X) = \frac{P(X, Y_1, Y_2)}{\sum_X P(X, Y_1, Y_2)} = \frac{P(X, Y_2 | Y_1)}{P(Y_2)}
\]

**Note:** This is so because, by independence of observations given \( X \), \( P(Y_1, Y_2 | X) = P(Y_1 | X) P(Y_2 | X) \)

**So we can derive** \( P(Y_2 | X) = P(Y_2, Y_1 | X) \)

**How?**

\[
P(Y_2 | X) \cdot P(Y_1 | X) = P(Y_1, Y_2 | X) = P(Y_1, Y_2 | X) P(Y_2 | X)
\]

**Similarly**

\[
P(X, Y_1, \ldots, Y_n | X) = \frac{P(X, Y_1, \ldots, Y_n)}{\sum_X P(X, Y_1, \ldots, Y_n)}
\]

\[
P(X | Y_1, \ldots, Y_n)
\]
limitations of this example:

If \( x \) is the state (say position) of an object being tracked by
saying \( x \) is stationary means the object is just sitting there not moving.
That is rather boring,

let's think now about a sequence on time series of states
\( x_1, x_2, \ldots, x_n \) that potentially have different values.
There are lots of ways to model how these values might depend on
each other.

Graphical model:

\[ \begin{array}{c}
\vdots \\
x_n \\
\vdots \\
x_1 \\
\end{array} \]

They could all be independent. This is
weird though, for a state of something
that we are tracking, note that the
\[ P(x_{\text{seq}}) = P(x_1) P(x_2) \cdots P(x_n) \]
could permute a set of indep variables and not change the
underlying statistical problem. This definitely does not
make sense for us.

By the way, to be concrete, if state vector \( x_2 = \begin{bmatrix} x_1 \\ 3 \end{bmatrix} \)
of 2D object position in an image, we can think of the
sequence of states \( x_1, \ldots, x_n \) as being a trajectory of the
object in the image.

At the other extreme, we could say every state \( x_k \) depends on
all the states that came before it.

\[ P(x_1, x_2, \ldots, x_n) = P(x_2) P(x_3|x_1) P(x_4|x_2) P(x_5|x_3) \cdots \]

This is completely general.

But, starts to become intractable computationally as our
trajectories get longer \& longer.
A common (almost ubiquitous) simplification is to make a first-order Markov assumption

\[ P(x_t | x_{t-1}, x_{t-2}, \ldots, x_1) = P(x_t | x_{t-1}) \]

"current state only depends on the previous state"

If a graphical model, we get a nice "chain"

\[ x_1 \rightarrow x_2 \rightarrow x_3 \rightarrow x_4 \]

and our joint distribution consists at most of pairwise functions

\[ P(x_1, \ldots, x_n) = P(x_1) P(x_2 | x_1) P(x_3 | x_2) \cdots P(x_n | x_{n-1}) \]

Note, for clarity: we are not talking at all yet about observations! \( x_1 \ldots x_n \) is an abstract ground truth trajectory, and \( \text{PC}(x_1 \ldots x_n) \) thus specifies our prior assumptions about what constitutes a "good" or likely trajectory. That is, I give you \( n \) points \((x_i, y_i)\) in a sequence, and you evaluate a function \( P(x_1 \ldots x_n) \) that says how likely it is that the trajectory I gave you looks like something that our object would have produced.

It encodes our "motion model" or "state dynamics".

Some info that can go into our function:
- objects tend to travel in a straight line (with constant velocity)
- maximum speed of travel
- trajectories tend to be smooth and continuous

Note: There is a trade-off between what we can represent with only pairwise (Markov) assumption \( P(x_t | x_{t-1}) \) vs what we include as an influence in the state vector.
For example, if our state is only 2D object position \((x_i)\) and we want to say the object at the next time step is computed using a constant velocity assumption, we could say something like:

\[
\begin{align*}
X_{k+1} &= X_k + (X_{k-1} - X_{k-2})
\end{align*}
\]

**But this violates the 1st-order Markov assumption** (\(X_k\) as a function of \((X_{k-1}, X_k, X_{k-2})\)) so we could use a second order Markov chain to deal with this, and that would be fine.

Or, if we want to stick with a 1st-order Markov chain, we have to augment the state vector with some extra variables that represent velocity, so state becomes \([x_i, 3; \dot{x}_i, \dot{3}]\) position 4D state vector.

We will talk more about this later.

Back to \(P(x_1, x_2, \ldots, x_n)\)

rather than a probability of a whole trajectory, maybe we just care about what the current state of the object, \(x_n\) is.

(for example, where is it now?)

We can find that by marginalizing out all the other variables

\[
P(x_n) = \sum_{x_1} \sum_{x_2} \ldots \sum_{x_{n-1}} P(x_1, x_2, \ldots, x_n) \, dx_1 \, dx_2 \ldots dx_{n-1}
\]
Our factored form (based on Markov assumption) makes it easy to carry this out as a nested set of computations:

\[
S \cdot S \cdot \ldots \cdot S \prod_{i=1}^{n} p(x_i | x_{i-1}) p(x_{i-1}) \ldots p(x_2 | x_1)
\]

This is completely equivalent to just recursively moving our way down the chain...

\[
p(x_1, x_2) = p(x_1 | x_2) p(x_2)
\]

\[
p(x_2) = \int_{x_1} p(x_1, x_2) \, dx_1
\]

\[
p(x_3) = \int_{x_2} p(x_2, x_3) \, dx_2
\]

\[
\vdots
\]

\[
p(x_{n-1}) p(x_n | x_{n-1}) = p(x_{n-1}, x_n)
\]

\[
p(x_n) = \int_{x_{n-1}} p(x_{n-1}, x_n) \, dx_{n-1}
\]

Note: There is no Bayesian rule going on here at all! Don't confuse this with a recursive Bayes estimation!

Note: "blind"
This is just pure prediction at worse! Given a prior estimate of object state \( x_1 \), we are crumbling through our motion prediction equations to predict where the object might be [actually, getting a distribution over where it might be] at the end of a trajectory.
Now, let's use our prior and make some observations! Assume \( Y_1, Y_2, \ldots, Y_n \) are noisy observations of states \( x_1, x_2, \ldots, x_n \). We then have

We now can use Bayes' rule to try to infer likely trajectories based on not just on our prior knowledge, but also on the measured data.

\[
P(x_1, x_2, \ldots, x_n) \quad \text{prior}
\]
\[
h(y_1, y_n | x_1 \ldots x_n) \quad \text{likelihood}
\]

\[
P(x_1 \ldots x_n | y_1 \ldots y_n) = \frac{P(x_1 \ldots x_n) h(y_1 \ldots y_n | x_1 \ldots x_n)}{\sum_{x_1 \ldots x_n} P(x_1 \ldots x_n) h(y_1 \ldots y_n | x_1 \ldots x_n)}
\]

Note, however, that this is a posterior over a whole trajectory. We may just care about a posterior over the current state \( x_n \). So again, we marginalize out \( x_1 \ldots x_{n-1} \).

\[
P(x_n | y_1 \ldots y_n) = \sum_{x_1 \ldots x_{n-1}} \sum_{x_n} P(x_1 \ldots x_n) h(y_1 \ldots y_n | x_1 \ldots x_n) dx_1 \ldots dx_{n-1}
\]

\[
\sum_{x_1 \ldots x_{n-1}} \sum_{x_n} P(x_1 \ldots x_n) h(y_1 \ldots y_n | x_1 \ldots x_n) dx_1 \ldots dx_{n-1}
\]

Note: This is a very general form of the Bayes filter for tracking. It is also nearly impossible to compute, so we have to appeal to our simplifications in order to tackle it (and to derive a recursive form for computation).
Our simplifications (independence assumptions)

Markov \( P(X_1, \ldots, X_n) = P(X_1) P(X_2 | X_1) \cdots P(X_n | X_{n-1}) \)

measurements only depend on current state \( P(Y_1 | X_1, X_2, \ldots, X_n) = P(Y_1 | X_1) P(Y_2 | X_2) \cdots P(Y_n | X_n) \)

We draw as the graphical model

\[
\begin{align*}
& X_1 \quad X_2 \quad X_3 \quad X_4 \quad \cdots \text{True states} \\
& Y_1 \quad Y_2 \quad Y_3 \quad Y_4 \quad \cdots \text{Observed data} \\
\end{align*}
\]

\[
P(X_1, \ldots, X_n, Y_1, \ldots, Y_m) = P(X_1, \ldots, X_n) P(Y_1, \ldots, Y_m | X_1, \ldots, X_n)
\]

\[
= P(X_1) P(Y_1 | X_1) P(X_2 | X_1) P(Y_2 | X_2) P(X_3 | X_2) P(Y_3 | X_3) \quad \cdots
\]

This is again a nice chain, containing both motion prediction and data-driven terms, interleaved.
This leads to a nice recursive estimation for \( P(X_1, \ldots, X_n) \) that interleaves two stages of motion prediction and data-driven correction.

**Initialization:** \( P(X_1) P(Y_1 | X_1) = P(X_1, Y_1) \)

\[
\frac{P(X_1, Y_1)}{P(X_1 | Y_1)} = \frac{P(X_1, Y_1)}{P(Y_1)}
\]

\[
\frac{\tilde{p}(X_1 | Y_1)}{P(X_1 | Y_1)} = \frac{P(X_1 | Y_1)}{P(Y_1)}
\]
motion prediction:
\[ P(x_{2|1}, x_1) P(x_1|y_1) = P(x_1, x_{2|1} | y_1) \]
\[ \sum_{x_1} P(x_{2|1}, x_1 | y_1) = P(x_2 | y_1) \] 

*at Time 2*

**Interpretation:** This is our prediction of the value of state \( x_2 \) based only on observations up to Time 1.

**Correction:** (recall, \( y_{2|2} \) also \( \rightarrow \) \( p(y_2 | x_{2|1}, y_1) \))
\[ P(x_{2|1} | y_1) P(y_2 | x_{2|1}) = P(x_{2|1}, y_2 | y_1) \]
\[ \frac{P(x_{2|1}, y_2 | y_1)}{P(y_2 | y_1)} = \frac{P(x_{2|1} | y_1) P(y_2 | y_1)}{P(y_2 | y_1)} = P(x_{2|1} | y_1) \]
\[ \sum_{x_{2|1}} P(x_{2|1} | y_1) P(y_2 | x_{2|1}, y_1) = P(x_{2|1} | y_1) \]

**Interpretation:** This is now our data-corrected estimate of the value of state at Time 2, based on all observations up to and including Time 2.

In general, say we have \( P(x_{k-1} | y_1 \ldots y_{k-1}) \)

**motion prediction step:**
\[ P(x_k | y_1 \ldots y_{k-1}) = \sum_{x_{k-1}} P(x_{k-1} | x_{k-1}) P(x_{k-1} | y_1 \ldots y_{k-1}) \]

**data correction step:**
\[ P(x_k | y_1 \ldots y_k) = \frac{P(y_k | x_k) P(x_k | y_1 \ldots y_{k-1})}{\sum_{x_k} P(y_k | x_k) P(x_k | y_1 \ldots y_{k-1})} \]

Next time & linear dynamic systems? All distributions are Gaussian?