

Intro to the Kalman filter

ISTA 410 / INFO 510: Bayesian Modeling and Inference

U. of Arizona School of Information

November 8, 2021

Last week:

- Covariance kernels and Gaussian process regression

This week:

- Hidden state models for sequential data
 - Kalman filter
 - Hidden Markov model

Temporal and dynamical models

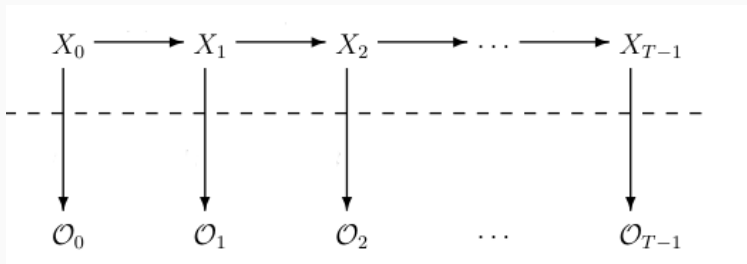
The models we'll look at next are used to model sequential data, especially time series.

- Hidden Markov model: latent state variables evolve according to a Markov chain/Markov process
- Linear dynamical system: latent state variables evolve according to linear dynamics, possibly with added noise

What these have in common: hidden/latent state variables

Hidden state models

We're going to focus on hidden state models, which have a general structure similar to below:



- Latent/hidden system state
- Observations based on system state

Two state-observation models

Two common models:

- Hidden Markov model

Hidden states evolve according to a Markov process

Observations typically Gaussian or multinomial

- Linear Gaussian dynamical system

States evolve according to linear dynamics

Observations a linear function of the state, “corrupted” by Gaussian noise

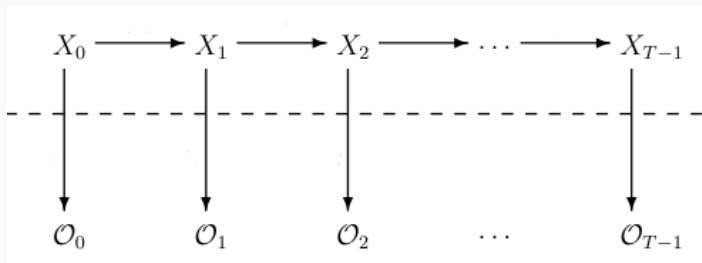
Typical inference problems

Typical problems we want to solve, given a sequence of observations \mathcal{O} of time length T :

- Filtering: find the distribution of X_T – that is, the distribution of the current state, accounting for all observations up to now.
- Prediction: find the distribution of X_t for some $t > T$.
- Smoothing: find the distribution of X_t for some $1 \leq t < T$.
This looks very similar to filtering, but differs in that we can take the observations after time t into account.
- MAP or best-explanation: find the sequence (X_i) maximizing $P(\mathcal{O}, X)$.
- Fitting: Given a sequence of observations, estimate the parameters of the underlying dynamical model.

Linear dynamical systems and the Kalman filter

Noisy linear dynamical systems



Specific structure:

- arrows are linear transformations with noise added

Reference: Welch & Bishop, *An Introduction to the Kalman Filter*
(on D2L)

Noisy linear dynamical systems

Basic system structure:

- Sequence of *states* x_k evolves according to a linear difference equation:

$$x_k = Ax_{k-1} + w_k$$

where w_k is a random noise term

- Sequence of observations z_k is a linear function of the state:

$$z_k = Hx_k + v_k$$

again, v_k is a noise term

Assumption: $w_k \sim N(0, Q)$, $v_k \sim N(0, R)$.

Here, N means multivariate normal.

Simplifying assumptions

Normality assumption for noise plays a big role:

$$w_k \sim N(0, Q)$$

$$v_k \sim N(0, R)$$

- Linear function of a normal RV is also normal
- Begin with normal prior on x_k : all states and observations are also normal
- Just need to estimate a mean vector (estimate) and covariance matrix (uncertainty/error)

Nonlinear dynamics: more complicated, sometimes can be approximated by a linear filter.

A Bayesian estimation step

The Kalman filter equations are based on an iterative application of Bayes' theorem.

Assume that we have, by magic, a prior estimate of the k th state, \hat{x}_k^- and this estimate's error covariance P_k^- , then, the distribution of the true x_k is:

$$x_k \sim N(\hat{x}_k^-, P_k^-)$$

Conditional on x_k , the distribution of the measurement z_k is

$$z_k | x_k \sim N(Hx_k, R)$$

Finally, the marginal distribution of the measurement z_k is

$$z_k \sim N(H\hat{x}_k^-, HP_k^- H^T + R)$$

A Bayesian estimation step

Plugging everything into Bayes' theorem, we get the posterior density for x_k .

$$p(x_k|z_k) = \frac{N(\hat{x}_k^-, P_k^-)N(Hx_k, R)}{N(H\hat{x}_k^-, HP_k^- H^T + R)}$$

A Bayesian estimation step

Plugging everything into Bayes' theorem, we get the posterior density for x_k .

$$p(x_k|z_k) = \frac{N(\hat{x}_k^-, P_k^-)N(Hx_k, R)}{N(H\hat{x}_k^-, HP_k^- H^T + R)}$$

With a bunch of ✨algebra✨ we can calculate the mean of x_k :

$$\hat{x}_k^- + (P_k^- H^T (HP_k^- H^T + R)^{-1})(z_k - H\hat{x}_k^-)$$

The highlighted factor is often denoted K_k , the *Kalman gain* or *blending factor*.

We also have the posterior covariance:

$$P_k = (I - K_k H)P_k^-$$

Posterior mean for the k th state is

$$\hat{x}_k^- + \underbrace{(P_k^- H^T (H P_k^- H^T + R)^{-1})}_{K_k} (z_k - H \hat{x}_k^-)$$

- prior estimate x_k^- plus a correction based on $(z_k - H \hat{x}_k^-)$
- $(z_k - H \hat{x}_k^-)$ is the residual (disagreement between observation z_k and predicted observation $H \hat{x}_k^-$).

Kalman gain determines the weight of each part.

This is best understood in two separate limits.

Role of the Kalman gain

If our measurements are really good, we should trust the observed values.

$$P_k^- H^T (H P_k^- H^T + R)^{-1} \rightarrow H^{-1} \text{ as } R \rightarrow 0$$

Then, in the limit of perfect measurement, the posterior mean is

$$\hat{x}_k^- + (P_k^- H^T (H P_k^- H^T + R)^{-1})(z_k - H \hat{x}_k^-) = H^{-1} z_k$$

(Remember H is the linear transformation mapping x_k to z_k .)

Role of the Kalman gain

The “opposite” limit occurs when the prior estimate’s error covariance goes to 0:

$$P_k^- H^T (H P_k^- H^T + R)^{-1} \rightarrow 0 \text{ as } P_k^- \rightarrow 0$$

in which case the posterior mean becomes

$$\hat{x}_k^- + (P_k^- H^T (H P_k^- H^T + R)^{-1})(z_k - H \hat{x}_k^-) = \hat{x}_k^-$$

The filtering algorithm

This assumed we had a prior estimate for the state and error covariance.

Where do we get these?

The filtering algorithm

This assumed we had a prior estimate for the state and error covariance.

Where do we get these?

From the previous time step! If we have estimates \hat{x}_{k-1} , P_{k-1} , the dynamical system gives:

$$\begin{aligned}\hat{x}_k^- &= A\hat{x}_{k-1} \\ P_k^- &= AP_{k-1}A^T + Q\end{aligned}$$

- take posterior estimates from step $k - 1$
- push them through the dynamics to get priors for step k
- adjust by incorporating the measurement

The filtering algorithm

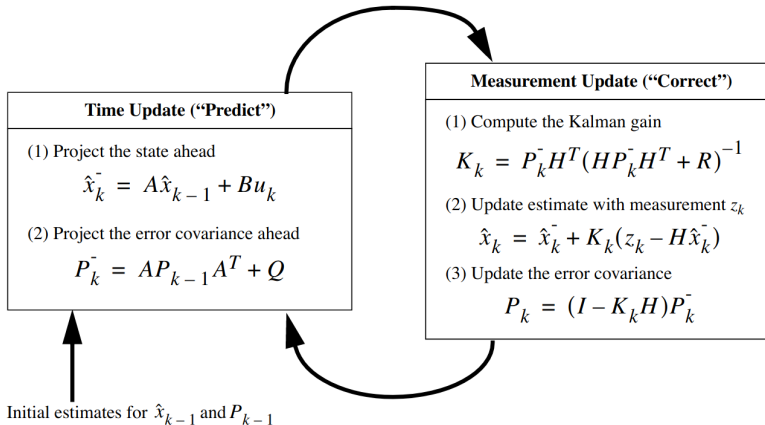
So, after setting initial estimates for \hat{x}_0^-, P_0^- , the filtering algorithm proceeds in two steps:

1. Obtain prior estimates \hat{x}_k^-, P_k^- by applying the dynamical system to \hat{x}_{k-1}, P_{k-1} . (“predict” step)
2. Compute the Kalman gain K_k and adjust estimates to \hat{x}_k, P_k . (“correct” step)

Notes:

- We’re taking advantage of normality and linearity here; all conditional and marginal distributions remain normal, so we can work only in terms of the mean/covariance.
- If Q, R are constant, then the estimate error covariance P_k and the gain K_k stabilize quickly and then stay constant.

The filter equations in a diagram



Parameters and tuning

A couple of comments on choosing the parameters Q, R :

- Measuring R empirically is usually practical, because it's a property of our measurement
- Q is trickier. Can come from a scientific model (ideally).
- Q can “compensate” for a poor process model by adding more uncertainty to state estimates – if we don't really know the dynamics, treat them as noise and the filter can still estimate states

Initial values of \hat{x}_0, P_0 not so important if the filter will run for some time.

Simplest example

Simplest possible example: estimating a constant

A very simple example: estimation of an unknown constant from a sequence of observations

For example: we are trying to estimate a voltage, but our instruments are faulty, introducing an amount of noise to each measurement.

This implies the following parameters:

- $A = 1$ (no deterministic time evolution of states)
- $H = 1$ (measuring voltage directly)
- $Q \approx 0$ (assume negligible fluctuation in states)
- $R = R_0$ (fixed measurement error)

Kalman filter equations

We can write down the Kalman filter equations for this version easily:

Dynamics update step:

$$\hat{x}_k^- = \hat{x}_{k-1}$$

$$P_k^- = P_{k-1} + Q$$

We can in principle set $Q = 0$, but we can also adjust it to allow for some fluctuations in the true voltage.

Kalman filter equations

The correction step:

$$K_k = \frac{P_k^-}{P_k^- + R}$$

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - \hat{x}_k^-)$$

$$P_k = (1 - K_k)P_k^-$$

Initial parameters

We need to pick values for:

- Q
- \hat{x}_0
- P_0
- R

We need to pick values for:

- Q : we'll use 10^{-5} for something very small but nonzero.
- \hat{x}_0 :
- P_0 :
- R :

We need to pick values for:

- Q : we'll use 10^{-5} for something very small but nonzero.
- \hat{x}_0 : we'll start with 0
- P_0 :
- R :

Initial parameters

We need to pick values for:

- Q : we'll use 10^{-5} for something very small but nonzero.
- \hat{x}_0 : we'll start with 0
- P_0 : this one determines how quickly we converge to a stable estimate – we'll try a few examples
- R :

Initial parameters

We need to pick values for:

- Q : we'll use 10^{-5} for something very small but nonzero.
- \hat{x}_0 : we'll start with 0
- P_0 : this one determines how quickly we converge to a stable estimate – we'll try a few examples
- R : this determines how much we “trust” the noisy measurements – we'll try a few examples

Let's try it...

A slightly less trivial example

Kalman filter with control

Kalman initially developed the filter for applications in control theory. So, alternate form:

$$x_k = Ax_{k-1} + Bu_{k-1} + w_k$$

(states)

$$z_k = Hx_k + v_k$$

(measurements)

- u_{k-1} , the control input, represents some linear forcing we can do to the system
- B defines how the control input influences the system

Modifying the filter equations

It turns out the only modification we need to make is to the prediction step:

$$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_{k-1}$$

$$P_k^- = AP_{k-1}A^T + Q$$

A toy example: position tracking for a robot

Imagine we're trying to track the position of a robot that moves through the following idealized physical environment:

- 2 dimensions (x/y or lat/long)
- No friction
- Fluctuations/turbulence \rightarrow white noise added to velocity

Our robot has four thrusters that allow us to apply a constant force in any of four directions

Suppose we also get periodic measurements of x/y position (quite noisy) and velocity (not so noisy)

Setting up the dynamics

We need four variables:

- x, y – position coordinates
- \dot{x}, \dot{y} – velocities

We'll use discrete time with a time step Δt .

Setting up the dynamics

If we assume no control, what is the dynamical relationship?

$$x_{k+1} = Ax_k$$

with

$$A = \begin{pmatrix} 1 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

We'll assume our control vectors look like

$$u = \begin{pmatrix} T_x \\ T_x \\ T_y \\ T_y \end{pmatrix}$$

where $T_x, T_y \in \{-1, 0, 1\}$ give the thruster state (+/- thrust or off), and that the thrusters produce a constant acceleration.

This gives rise to the B matrix:

$$B = \begin{pmatrix} \frac{1}{2}k\Delta t^2 & 0 & 0 & 0 \\ 0 & k\Delta t & 0 & 0 \\ 0 & 0 & \frac{1}{2}k\Delta t^2 & 0 \\ 0 & 0 & 0 & k\Delta t \end{pmatrix}$$

This gives rise to the B matrix:

$$B = \begin{pmatrix} \frac{1}{2}k\Delta t^2 & 0 & 0 & 0 \\ 0 & k\Delta t & 0 & 0 \\ 0 & 0 & \frac{1}{2}k\Delta t^2 & 0 \\ 0 & 0 & 0 & k\Delta t \end{pmatrix}$$

This gives us enough to implement the model.

Kalman filters assume linear dynamical systems, but many systems are nonlinear

Option 1: extended Kalman filter

- Linearize the dynamical system around the estimate at each time step
- Advantage: pretty simple, same ideas work
- Disadvantages:
 - Assumptions of normality no longer hold
 - No longer really Bayesian – just an approximation
 - Errors can accumulate over time

Other approaches, based on sampling:

- Ensemble Kalman filter: run a large number of Kalman filters, with added noise, and average the results
- Unscented Kalman filter: in the update step, project forward a deterministically chosen set of points to estimate the mean/variance at the next time step
- Particle filter: use importance sampling to maintain a sample of state sequences, pruning and splitting them at each time step

Today:

- Discrete-time Kalman filter

Next:

- More time series filtering
- Hidden Markov models