Lecture 26: Multivariate Kalman filtering

Winter 2015

Ref: Hartmann, Ch. 8

26.1 Tracking a ball

We're playing center field in a baseball game. The batter hits the ball toward us. We need to quickly judge where it is going to land, so we can run and catch it.

We can frame this as a sequential estimation problem. Let (r_n, z_n) be the true position of the ball with respect to us at successive times $t = (n-1)\Delta t$. At each time, we observe the angle of the ball

$$y_n = \tan^{-1}(z_n/r_n)$$

with some error. Our mathematical model for the ball motion is Newton's laws for an object moving in a gravitational field. These require that at each time we also estimate the ball's horizontal and vertical velocity components u_n and w_n.

For this system, there are four variables to estimate, with one observation at each time of a quantity that is a nonlinear function of the unknowns. This motivates us to extend Kalman filtering to more complex systems.

26.2 Multivariate, nonlinear systems

The analysis of this system is more complex than the simple case, because

1. We are estimating multiple variables at each time n.
2. We are not directly observing the full state at each time.
3. Errors in the state variable estimates will be correlated with each other.
4. The measured quantity is a nonlinear function of the state.

The Kalman filtering strategy is to use the new observations to update the estimated multivariate state $\mathbf{x}$ and its covariance matrix at each time n. We derive general mathematical machinery for handling multivariate, nonlinear problems given sequential observations of quantities that depends on the state.

26.3 Sequential estimation roadmap for multivariate problems

Consider a possibly nonlinear system of the form

$$\mathbf{x}_n = f(\mathbf{x}_{n-1}). \tag{26.3.1}$$

where the state $\mathbf{x}$ has m components. We will not include noise in the model itself, though this is easy to add and is traditionally included in the theory of Kalman filtering.

At each time n, we get a new set of q observations $\mathbf{y}$ of quantities $\mathbf{h}(\mathbf{x})$ that depend (possibly nonlinearly) on the state, with observation errors $\mathbf{y}'$ that have a $q \times q$ covariance matrix $\mathbf{C}^o$. If these errors are independent, this covariance matrix will be diagonal, with the variances of the observations as the diagonal elements. Note that q needn't be equal to m. If $q < m$ the new measurements by themselves would underdetermine the state ; if $q > m$ they would overdetermine the state.

As in the 1D case, we break the sequential estimation problem into steps:

Initialize Make an initial estimate of the model state $\mathbf{x}$ and of the matrix of error covariances $\mathbf{C}$ between the estimated state variables. There is no one right way to do this, but after a few time steps the algorithm will have blended in enough observations to make it fairly insensitive to the the initialization details.

Predict Using the model equations, and the estimated state $\hat{\mathbf{x}}_{n-1}$ and covariance matrix $\hat{\mathbf{C}}_{n-1}$ from time step $n-1$, make predictions $\mathbf{x}_n^p$ and $\mathbf{C}_n^p$ of these quantities at time n.

Update Combine these predictions with the new observations at time n to update the estimated state $\hat{\mathbf{x}}_n$ and covariance matrix $\hat{\mathbf{C}}_n$. Then cycle back to step 2.

We will state (but not derive) the prediction and update steps for a general nonlinear multivariable system, which can be written in a form parallel to our 1D case. It involves two additional matrices.

The **update matrix** $\mathbf{F}_{n-1}$ $[m \times m]$ is derived by linearizing the model function f used to predict time n from the estimated state at time $n-1$:

$$\mathbf{x}'^p = \mathbf{F}_{n-1}\hat{\mathbf{x}}'_{n-1} \tag{26.3.2}$$

where a prime denotes an arbitrary small uncertainty in the state. This is sometimes called the *linear tangent model*. $\mathbf{F}_{n-1}$ is calculated as the *Jacobian matrix* of the nonlinear function f, whose ij'th component is $\partial f_i/\partial x_j(\hat{\mathbf{x}}_{n-1})$, where f_i is the i'th component of the vector-valued function f. The Jacobian can be calculated analytically if f is simple or approximated numerically if it is not.

The **measurement matrix** $\mathbf{H}_n$ $[q \times m]$ relates the prediction uncertainty $\mathbf{y}'^p$ in the measurements to the prediction uncertainty $\mathbf{x}'^p$ in the state. It is derived by linearizing (if needed) the measurement function h about the predicted state at time n:

$$\mathbf{y}' = \mathbf{H}_n \hat{\mathbf{x}}' \tag{26.3.3}$$

It can again be computed as a Jacobian matrix.

The Kalman filtering algorithm is a sequence of linear algebra steps:

Simple 1D	General Kalman filter
Predict	Predict
$x_n^p = a\hat{x}_{n-1}$	$\mathbf{x}_n^p = f(\hat{\mathbf{x}}_{n-1})$
$\sigma_p^2 = a^2\hat{\sigma}_{n-1}^2$	$\mathbf{C}^p = \mathbf{F}_{n-1}\hat{\mathbf{C}}_{n-1}\mathbf{F}_{n-1}^T$
Update	Update
$\hat{x}_n = x_n^p + k(x_n^o - x_n^p)$	$\hat{\mathbf{x}} = \mathbf{x}^p + \mathbf{K}(\mathbf{y}^o - \mathbf{h}(\mathbf{x}^p))$
$k = \sigma_p^2/(\sigma_p^2 + \sigma_o^2)$	$\mathbf{K} = \mathbf{C}^p\mathbf{H}_n^T(\mathbf{H}_n\mathbf{C}^p\mathbf{H}_n^T + \mathbf{C}^o)^{-1}$
$\hat{\sigma}_n^2 = (1-k)\sigma_p^2$	$\hat{\mathbf{C}}_n = (\mathbf{I} - \mathbf{K}\mathbf{H_n})\mathbf{C}^p$

Here $\mathbf{C}^p$ $[m \times m]$ is the covariance matrix of the prediction at time n, $\hat{\mathbf{C}}$ $[m \times m]$ is the covariance matrix of the update, $\mathbf{C}^o$ $[q \times q]$ is the covariance matrix of the observations, and $\mathbf{K}$ $[m \times q]$ is the Kalman gain matrix.

It is mathematically involved to derive all the steps in the Kalman filtering algorithm, so we just admire the close analogy to the single-variable case, and go on to the ball-tracking example. In practice, as with optimal interpolation, the practical issue in applying this to models with lots of predictive variables (such as weather forecast models) is that it involves inversion of full high-dimensional matrices.

The **ensemble Kalman filter** (EnKF) uses an ensemble of model simulations, each initialized differently but forced with identical observations, to estimate the error covariance matrix of the model state, rather than trying to compute it from the model equations. A group led by Jeff Anderson at NCAR has developed flexible software called DART based on EnKF to assimilate data into a broad class of predictive models (https://www.image.ucar.edu/DAReS/DART)

26.4 Ball-tracking example: Model equations

Imagine we are tracking a batted ball moving under gravity. Let (r_n, z_n) be the horizontal and vertical components of the true position of the ball at successive times $t = (n-1)\Delta t$, and let (u_n, w_n) be the corresponding velocity components.

The **model equations** are

$$\begin{aligned} r_n &= r_{n-1} + u_{n-1}\Delta t \\ z_n &= z_{n-1} + w_{n-1}\Delta t - g\Delta t^2/2 \\ u_n &= u_{n-1} \\ w_n &= w_{n-1} - g\Delta t, \end{aligned}$$

where $g = 9.8$ m s^{-2} is the downward acceleration of gravity. Defining the **state**

$$\mathbf{x} = [r\ z\ u\ v]^T,$$

this defines the **model function** $\mathbf{x}_n = f(\mathbf{x}_{n-1})$. Because f is a linear function of the $m = 4$ unknowns, it is easy to deduce the 4×4 **update matrix**

$$F = \begin{bmatrix} \partial f_1/\partial r & \partial f_1/\partial z & \partial f_1/\partial u & \partial f_1/\partial v \\ \partial f_2/\partial r & \partial f_2/\partial z & \partial f_2/\partial u & \partial f_2/\partial v \\ \partial f_3/\partial r & \partial f_3/\partial z & \partial f_3/\partial u & \partial f_3/\partial v \\ \partial f_4/\partial r & \partial f_4/\partial z & \partial f_4/\partial u & \partial f_4/\partial v \end{bmatrix} = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

26.5 Observations

At each time, we observe the angle of the ball

$$y_n = \tan^{-1}(z_n/r_n)$$

with variance σ_o^2. Hence, the **measurement function** is

$$y = h(\mathbf{x}) = \tan^{-1}(z/r).$$

From the Jacobian of this function we get the 1×4 **measurement matrix**,

$$H = \begin{bmatrix} \partial h/\partial r & \partial h/\partial z & \partial h/\partial u & \partial h/\partial v \end{bmatrix} = \begin{bmatrix} \frac{-z}{r^2+z^2} & \frac{r}{r^2+z^2} & 0 & 0 \end{bmatrix}.$$

Since there is $q = 1$ measurement per time step, the 1×1 **measurement covariance matrix** is

$$C^o = [\sigma_o^2].$$

26.6 Initialization

We know the location $r_0, z_0 = 0$ at which the bat hit the ball with small uncertainty but we have to arbitrarily guess the ball's initial velocity u_0, v_0, and hence assign a large uncertainty to them. We use these principles to calculate the initial state vector $\hat{\mathbf{x}}_1$ and covariance matrix $\hat{\mathbf{C}}_1$. The hope is that the observations will quickly constrain the velocities and the covariance matrix. This is not assured, because we only have one measurement and four variables.

26.7 Results

The Matlab code **kalman2** implements a Kalman filter based on the above, with plausible numerical values chosen for all the parameters and initial conditions. Within 12 observations (1.2 second) the approach converges fairly accurately to the true solution. Interestingly, the predicted variance of the ball position (especially height) increases over most of the ball trajectory, but seems to overestimate the actual error. The ball landing location is well predicted based on the estimated state at all times more than 1.2 seconds after it is batted.

26.8 Limitations of the Kalman filtering approach

Kalman filtering is a powerful method for keeping a model of some process optimally matched in real time to a sequence of noisy observations. However, it does have some important pitfalls and limitations:

Computational complexity If the model has many predictive variables, and especially if there are also observations of many quantities at each time (e. g. a global weather forecast), the prediction and update of the covariance matrix can become computationally unaffordable, since it involves extremely large matrices. Shortcuts to simplify this process have their own problems.

Overconfidence The method works best if the variances of the observations and of the predicted values of the measurements stay comparable. With long time steps between updates or with incomplete or very noisy observations, the predicted model uncertainty can become much less than the observational uncertainty in some variables, which will cause the updated state to nearly ignore those observations and drift away from reality. To combat this, it is often necessary to inflate the variance of the model uncertainty in some way, e. g. by adding artificial noise into the model.