

The Kalman filter

The Kalman filter is a commonly used technique in signal processing; it is one of the main tools used in the guidance and control of autonomous agents. We are interested in it here because it is another example of Bayesian inference which is actually somewhat similar to the sensory fusion example we saw earlier. It is probably that Kalman filtering is implemented neurally; we will look at an example of that.

We will do the simplest example and introduce it in the case of a navigation along a track; obviously this is different the examples we might be interested in, but it makes the story easy to tell. Imagine someone cycling along a path; they have a GPS accurate to within 10m but for reasons we might make up, they want a more accurate estimate of their position. They walk at a constant rate, well nearly constant, it might vary a little bit from minute to minute because of gusts of wind, slight variations in slope. If they are moving at a constant rate, they have a second approach to calculating their position, that is by dead reckoning, multiplying their time of travel by their speed. The idea behind a Kalman filter is to make the optimal combination of these two noisy pieces of information about their position.

Lets write down some equations. Let

$$\bar{\mathbf{x}} = \begin{pmatrix} \bar{s} \\ \bar{v} \end{pmatrix} \quad (1)$$

represent the current estimate of the position, \bar{s} , and speed, \bar{v} , of the cyclist. This gets updated in two different ways; by dead reckoning and because of the measurement, using the GPS. In fact, what we mean by $\bar{\mathbf{x}}$ is that it is the mean of our probability distribution for position x and speed s . Throughout what follows, all distributions are Gaussian and we will benefit from the usual Gaussian magic whereby whenever you do stuff to Gaussians you get more Gaussians. In fact, the Kalman filter works well for other distributions, because distributions often look like Gaussians, but it is only proved here to be the optimal approach for Gaussian uncertainty. Of course, Gaussians are described by two sets of numbers, the means, as above, and the covariance matrix, that is, in this case the two dimensional version of the variance. As we work through the derivation, we will have recourse to the Gaussianity of the distributions, but the actual algorithm is described in terms of the means and variances.

Using the obvious notation, covariance is given by

$$P = [\langle (x_i - \bar{x}_i)(x_j - \bar{x}_j) \rangle] \quad (2)$$

The idea behind the Kalman filter is to update both the mean and covariance. This is done using Bayesian inference, but it is phrased in terms of the mean and covariance, rather than the full probabilities.

Lets consider dead reckoning first. Imagine the cyclist consults their GPS every δt . If there was no noise then $x \rightarrow x + v\delta t$ and, since the speed remains the same, $v \rightarrow v$. In fact, it is easy to include the possibility of variable acceleration by including a *control vector* to describe the cyclist's intentional changes in acceleration. Although this is a fairly simple addition, we won't include that here to keep the notation as clear as possible. Lets use a subscript a to mean the change of the position after movement

$$\mathbf{x}_a = F\mathbf{x} + \mathbf{w} \quad (3)$$

where F is the motion matrix

$$F = \begin{pmatrix} 1 & \delta t \\ 0 & 1 \end{pmatrix} \quad (4)$$

and \mathbf{w} represents zero mean Gaussian noise because of random variations in speed and position. Let the covariance matrix for \mathbf{w} be Q .

Now, take the average across equation for \mathbf{x} , this is easy because the Gaussian noise has zero mean, so we get

$$\bar{\mathbf{x}}_a = F\bar{\mathbf{x}} \quad (5)$$

This is our estimate of the position after motion based on dead reckoning. For convenience we will write $\mathbf{x}_d = \bar{\mathbf{x}}_a$

The change to P is more complicated. Lets motivate the result with a simpler example. Say we had a variable y with variance p so y is drawn from $\mathcal{N}(\bar{y}, p)$ and say that y is transformed by

$$y_d = fy + u \quad (6)$$

where f is some number, the analogue of the motion matrix and u is noise drawn from $\mathcal{N}(0, q)$. Calculating the new distribution is actually a bit complicated because it involves a convolution but lets just quote the result: among

the nice properties of Gaussians is that the convolution of Gaussians is a Gaussian. The nice thing is that if we know distribution for y_d is a Gaussian then we only need to calculate its moments and that is easy

$$\bar{y}_d = \langle y_d \rangle = \langle fy + u \rangle = f\langle y \rangle = f\bar{y} \quad (7)$$

and with some more algebra

$$\begin{aligned} \langle (y_d - \bar{y}_d)^2 \rangle &= \langle (fy + u)^2 \rangle - \bar{y}_d^2 = f^2 \langle y^2 \rangle + \langle u^2 \rangle - f^2 \bar{y}^2 \\ &= f^2(p + \bar{y}^2) + q - f^2 \bar{y}^2 = f^2 p + q \end{aligned} \quad (8)$$

where we have used the assumption that there is no correlation between the new noise u and uncertainty in y .

The actual change in P can be derived using similar algebra which you can check, we will just quote the result:

$$P_d = FPF^T + Q \quad (9)$$

where the superscripted T means transpose. This is our change in our estimate of our uncertainty in the position of the cyclist after the motion.

Now we have dealt with the estimate based on dead reckoning, we consider the sensor noise. Imagine the sensor measures the true position along with some noise

$$\mathbf{x}_s = \mathbf{x}_a + \mathbf{r} \quad (10)$$

where \mathbf{r} is the noise in our sensor; it has covariance R . The full model has a more complicated sensor noise model, it allows

$$\mathbf{x}_s = H\mathbf{x}_a + \mathbf{r} \quad (11)$$

for some matrix H , but for simplicity we'll ignore that here. Again, it doesn't make the calculation that much harder.

We now have two estimates of the position, one based on dead reckoning and the other based on the sensor. The challenge now is to work out the best estimate for the position. We want $p(\mathbf{x}_a | \mathbf{x}_d \mathbf{x}_s)$. From the Bayes rule:

$$p(\mathbf{x}_a | \mathbf{x}_d \mathbf{x}_s) \propto p(\mathbf{x}_d \mathbf{x}_s | \mathbf{x}_a) = p(\mathbf{x}_d | \mathbf{x}_a) p(\mathbf{x}_s | \mathbf{x}_a) \quad (12)$$

where we have made the usual sort of assumption of conditional independence. Now all we need to do is to multiply the Gaussians and find the new mean and variance. Obviously this is difficult because it is in two dimensions.

Lets look at a one dimensional version where we will use y wherever there is a \mathbf{x} and little letters for variances to correspond to all the big letter covariance matrices:

$$p(y_a|y_d y_s) \propto p(y_d|y_a)p(y_s|y_a) \quad (13)$$

where $P(y_d|y_a)$ corresponds to $\mathcal{N}(y_a, p_d)$ and $p(y_s|y_a)$ corresponds to $\mathcal{N}(y_s, r)$. Now we did just this calculation for sensory fusion, we know that we get another Gaussian, well up to a constant, with

$$\frac{1}{\sigma^2} = \frac{1}{p_d} + \frac{1}{r} \quad (14)$$

and this gives the new mean

$$y_n = \frac{\sigma^2}{p_d} y_d + \frac{\sigma^2}{r} y_s \quad (15)$$

These correspond to our new estimates for the position and variance. In other words they give new values of y and p . In fact, to match the way the Kalman filter equations, we rewrite this a bit using

$$\frac{\sigma^2}{p_d} = 1 - \frac{\sigma^2}{r} \quad (16)$$

so

$$y_n = y_d + k(y_s - y_d) \quad (17)$$

where

$$k = \frac{\sigma^2}{r} \quad (18)$$

Thus, the new estimate is the dead reckoning estimate along with a correction coming from the sensor.

Similarly

$$\frac{1}{\sigma^2} = \frac{1}{p_d} + \frac{1}{r} = \frac{p_d + r}{r p_d} \quad (19)$$

so

$$\sigma^2 = \frac{r p_d}{p_d + r} \quad (20)$$

and hence

$$k = \frac{p_d}{p_d + r} \quad (21)$$

with

$$\sigma^2 = rk \quad (22)$$

or, eliminating r

$$\sigma^2 = \frac{rp_d}{p_d + r} = \frac{p_d(r + p_d)}{p_d + r} - \frac{p_d^2}{p + d_r} = (1 - k)p_d \quad (23)$$

These particular combinations are used, or mentioned, because they relate to the notation usually used for the Kalman filter.

Now we need the two-dimensional version of all this. We won't derive the equations, but they are similar enough in spirit to the one-dimensional case to look like they should be correct, the actual derivation is just a matter of some, pretty tedious, algebra. We define

$$S = P_d + R \quad (24)$$

and

$$K = P_d S^{-1} \quad (25)$$

This factor, called the *Kalman gain* is clearly the analogue of k above. Now, it can be shown that

$$\mathbf{x}_n = \mathbf{x}_d + K(\mathbf{x}_s - \mathbf{x}_d) \quad (26)$$

and

$$P_n = (\mathbf{1} - K)P_d \quad (27)$$

where the bold one is the identity matrix. This gives the full update; \mathbf{x}_n and P_n are the basis for the next round of estimation. To describe the full algorithm we will add a time argument and let $t_{i+1} = t_i + \delta t$. The algorithm has three steps. First dead reckoning:

$$\begin{aligned} \mathbf{x}_d(t_{i+1}) &= F\mathbf{x}(t_i) \\ P_d(t_{i+1}) &= FP(t_i)F^T + Q \end{aligned} \quad (28)$$

Next there is the sensor giving us $\mathbf{x}_s(t_{i+1})$. From these we work out the new estimates

$$\begin{aligned} \mathbf{x}(t_{i+1}) &= \mathbf{x}_d(t_{i+1}) + K[\mathbf{x}_s(t_{i+1}) - \mathbf{x}_d(t_{i+1})] \\ P(t_{i+1}) &= (\mathbf{1} - K)P_d(t_{i+1}) \end{aligned} \quad (29)$$

Thus, in summary, the Kalman filter is a Bayesian optimal fusion of two sources of information: a dead reckoning estimate and an estimate coming

from a sensor; although it calculates distributions, it is written in terms of means and covariance matrices, this is sufficient since everything is assumed to be Gaussian. It gives an efficient and succinct method to track position. The substantial drawback is that it requires knowledge of both R and Q ; it is easy to envisage an estimate of R since the sensor noise could be estimated from repeated measurements at a fixed position, it is less clear how Q could be calculated.