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; it dates back to the 1960s and because it is a reasonably light-weight algorithm, it was crucial to space exploration. 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 say 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 ride 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. The key point is that, although the main aim is to estimate position, the estimate is done using Bayesian fusion, so the variance is also estimated.

One of the challenges in understanding Kalman filtering is that it was developed as a practical engineering solution to control and it often written in the language of control theory using terms and notations that are different to what we would use. However, this isn't the whole challenge with notation, it is difficult to avoid the notation becoming complicated. One issue is that it is an update equation, the goal is to update an estimate for position from at time t to another time $t + \delta t$ so it is tempting to have lots of t indices around which quickly clutter up the equations. To avoid that here I just look at one update, from a current estimate to an estimate one time step later.

Lets write down some equations. Let

$$\mathbf{x} = \begin{pmatrix} s \\ v \end{pmatrix} \tag{1}$$

represent the current knowlege of the position, s, and speed, s, of the cyclist. We believe these two values are drawn from a multivariate normal distribution, representing our uncertainty, so

$$\mathbf{x} \sim \text{MvNormal}(\mathbf{x}_e, P_e) \tag{2}$$

so our understanding of the position and speed is represented by two values, the mean $\mathbf{x}_e$ and covariance, P_e, of the two-dimensional normal: where, as ever, the mean is

$$\mathbf{x}_e = \langle \mathbf{x} \rangle = \bar{\mathbf{x}} \tag{3}$$

and, here, the covariance is the higher-dimensional generalization of the variance:

$$P_{ij} = [\langle (x_i - \bar{x}_i)(x_j - \bar{x}_j) \rangle \tag{4}$$

Kalman filtering is usually a multi-dimensional calculation, two-dimensional for tracking as here, higher-dimensional for more abstract estimation problems; that means working with multi-dimensional normal distributions; however, here, we will try to avoid any complex calculations in two dimensions.

In effect $\mathbf{x}_e$ is our estimate ofour position. This gets updated in two different ways; by dead reckoning and because of the measurement, using the GPS.

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 and this is what maks Kalman filtering so useful, it tracks knowledge using a probability distribution, but all the calculations are written in terms of the mean and covariance. Again, the idea behind the Kalman filter is to update both the mean and covariance. This is done using Bayesian inference, but ultimately it is phrased so the algorithm is writen in terms of the mean and covariance, rather than the full probability distribution.

Lets consider dead reckoning first. Imagine the cyclist consults their GPS every δt. If there was no noise then

$$s \to s + v\delta t \tag{5}$$

and, since the speed remains the same,

$$v \to v. \tag{6}$$

In fact, it is easy to include the possibility of variable accelleration by including a *control vector* to describe the cyclist's intentional changes in accelleration. 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

$$\mathbf{x}_a = F\mathbf{x} + \mathbf{w} \tag{7}$$

where F is the motion matrix

$$F = \begin{pmatrix} 1 & \delta t \\ 0 & 1 \end{pmatrix} \tag{8}$$

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:

$$\mathbf{w} \sim \text{MvNormal}(\mathbf{0}, Q) \tag{9}$$

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

$$\langle \mathbf{x}_a \rangle = F\langle \mathbf{x} \rangle = F\mathbf{x}_e \tag{10}$$

This is our estimate of the position after motion based on dead reckoning. For convenience we will write

$$\mathbf{x}_d = \bar{\mathbf{x}}_a \tag{11}$$

The equivalent covariance, P_d is more complicated: we need to calculate the covariance of the distribution for $\mathbf{x}_a$. Lets motivate the result with a simpler example in one-dimension; we will do this through these notes: rather than looking at the full two-dimensional problem, we will examine the one-dimensional case where the calculation is easier and then exhibit the two-dimensional equivalent, appealing to their similarity to the one-dimensional result as some evidence they are correct. The two-dimensional calculations are not difficult, it is more that it is easier to get insight into the main ideas by looking in one-dimension.

So say we had a variable y with variance p so y is drawn from $\text{Normal}(y_e, p)$ and say that y is transformed by 'motion', actually in this one-dimensional case just a scaling:

$$y_a = fy + u \tag{12}$$

where f is some scaling factor acting as an analogue of the motion matrix. u is noise drawn from $\text{Normal}(0, q)$. Calculating the new distribution is actually a bit complicated because it involves a convolution but lets just quote the result: among the useful properties of Gaussians is that the convolution of Gaussians is a Gaussian. The nice thing is that if we know distribution for y_a is a Gaussian then to describe it we only need to calculate its first two moments, its mean and variance, and that is easy

$$y_d = \langle y_a \rangle = \langle fy + u \rangle = f\langle y \rangle = fy_e \tag{13}$$

so the dead reckoning estimate of y is f times the previous mean. and with some more algebra

$$\begin{aligned} \langle (y_a - y_d)^2 \rangle &= \langle (fy + u)^2 \rangle - y_d^2 = f^2\langle y^2 \rangle + \langle u^2 \rangle - f^2 y_e^2 \\ &= f^2(p + y_e^2) + q - f^2 y_e = f^2 p + q \end{aligned} \tag{14}$$

where we have used the assumption that there is no correllation between the new noise u and uncertainty in y: note also that we have used

$$p = \langle y^2 \rangle - y_e^2 \tag{15}$$

which comes from expanding out the definition

$$p = \langle (y - y_e)^2 \rangle \tag{16}$$

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

$$P_d = FP_eF^T + Q \tag{17}$$

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

$$\mathbf{x}_s = \mathbf{x} + \mathbf{r} \tag{18}$$

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

$$\mathbf{x}_s = H\mathbf{x}_a + \mathbf{r} \tag{19}$$

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

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

$$p(\mathbf{x}_a|\mathbf{x}_d, \mathbf{x}_s). \tag{20}$$

From the Bayes rule:

$$p(\mathbf{x}_a|\mathbf{x}_d, \mathbf{x}_s) \propto p(\mathbf{x}_d\mathbf{x}_s|\mathbf{x}_a) = p(\mathbf{x}_d|\mathbf{x}_a)p(\mathbf{x}_s|\mathbf{x}_a) \tag{21}$$

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 whereever there is a $\mathbf{x}$ and little letters for variances to correspond to all the big letter covariance matrices:

$$p(y_a|y_d, y_s) \propto p(y_d|y_a)p(y_s|y_a) \tag{22}$$

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

$$\frac{1}{\sigma^2} = \frac{1}{p_d} + \frac{1}{r} \tag{23}$$

and this gives the new mean

$$y_n = \frac{\sigma^2}{p_d}y_d + \frac{\sigma^2}{r}y_s \tag{24}$$

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

$$\frac{\sigma^2}{p_d} = 1 - \frac{\sigma^2}{r} \tag{25}$$

so

$$y_n = y_d + k(y_s - y_d) \tag{26}$$

where

$$k = \frac{\sigma^2}{r} \tag{27}$$

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

Similarly

$$\frac{1}{\sigma^2} = \frac{1}{p_d} + \frac{1}{r} = \frac{p_d + r}{rp_d} \tag{28}$$

so

$$\sigma^2 = \frac{rp_d}{p_d + r} \tag{29}$$

and hence

$$k = \frac{p_d}{p_d + r} \tag{30}$$

with

$$\sigma^2 = rk \tag{31}$$

or, eliminating r

$$\sigma^2 = \frac{rp_d}{p_d + r} = \frac{p_d(r + p_d)}{p_d + r} - \frac{p_d^2}{p + d_r} = (1 - k)p_d \tag{32}$$

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

In summary we start with y_e and P_e; the mean and covariance describing our knowledge of the position and arrive at new values y_n and P_n for our knowledge after a time δt; these are the mean and covariance of our new estimates.

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

$$S = P_d + R \tag{33}$$

and

$$K = P_d S^{-1} \tag{34}$$

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

$$\mathbf{x}_n = \mathbf{x}_d + K(\mathbf{x}_s - \mathbf{x}_d) \tag{35}$$

and

$$P_n = (\mathbf{1} - K)P_d \tag{36}$$

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:

$$\begin{aligned} \mathbf{x}_d(t_{i+1}) &= F\mathbf{x}(t_i) \\ P_d(t_{i+1}) &= FP(t_i)F^T + Q \end{aligned} \tag{37}$$

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

$$\begin{aligned} \mathbf{x}(t_{i+1}) &= \mathbf{x}_d(t_{i+1}) + K[\mathbf{x}_s(t_{i+1}) - \mathbf{x}_d(t_{i+1})] \\ P(t_{i+1}) &= (\mathbf{1} - K)P_d(t_{i+1}) \end{aligned} \tag{38}$$

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. In the brain, the cerebellum may implement something like a Kalman filter to allow motion tracking during movement; more generally, this provides a paradigm for how prediction can be compared to sensory input, something that is central to the operation of the brain.