Kalman Filter

Alexandru Meterez

April 2020

1 Introduction

Kalman Filter works by combining two possible bad measurments on their own (with big errors) into a much better one:

- the physical model of the system
- the measurements of the sensors

Let's take an example. Let's say we have a car and we want to know its position. For this we have 2 sensors, one that measures the **position** and one that measures the **speed**. Unfortunately, both of them have errors. However, we can incorporate the measurements with the physical model of the system $x_n = x_{n-1} + v_{n-1} \cdot \Delta t$ in order to minimise the error. In other words, we combine:

- I know where I am know, my speed and the time and according to Newton, I'll be at some x_n later
- I know that my sensor tells me that I will be at some other x_n

Both of them are "wronger" on their own, then together.
A visual explanation is that it works based on overlapping Gaussians. You have a Gaussian around the estimate from the physical model and one around the measurement from the sensors (see figure 1). If you calculate the intersection (product of Gaussians), you get a smaller variance and thus a smaller error (see figure 2).

2 Algorithm

The Kalman Filter algorithm is pretty easy and it consists of 2 major steps:

1. **Predict** the state based on the physical system
2. **Update** the prediction based on the measurement from the sensors

We thus have an **iterative process**.

3 1D Kalman Filter

Predict

$$\hat{x_k} = \hat{x_{k-1}} + \Delta t \cdot \hat{\dot{x}_{k-1}} \tag{1}$$

$$\hat{\dot{x}_k} = \hat{\dot{x}_{k-1}} \tag{2}$$

$$p_k = p_{k-1} + q \tag{3}$$

Update

$$K_k := p_k (p_k + r)^{-1} \tag{4}$$

$$\hat{x_k} := \hat{x_k} + K_k (z_k - \hat{x_k}) \tag{5}$$

$$p_k := (1 - K_k) p_k \tag{6}$$

And now plugin x_k and p_k into the predict step again and iterate.
Observation: The equations (1) and (2) are not set in stone. This example is for the position of a moving object with constant velocity, that's why the rules look like this. Also, because I want them to look similar to the way they are written in the matrix form. But they can change based on what the system is doing.

- x_k - position at time k
- $\dot{x_k}$ - velocity at time k
- p_k - estimation error - how far our estimate is from the truth
- q - process error - how far our physical model is from the truth (for example it doesn't include wind, or wheel friction and so on)
- r - measurement error - how far our sensors are from the truth
- K_k - Kalman gain - looks at how big the errors of the estimate and the measurement are in eq. (4) and then chooses how much to take into consideration (weighs) the measurement and the estimate in eq. (5)
- z_k - measurement from sensors

This notation is cumbersome, especially if we want to have a stat with many more variables. We can go to the matrix form, by placing the position and speed in a vector and vectorize the equations.

4 Matrix form Kalman Filter

Predict

$$\hat{\boldsymbol{x_k}} = F_k \hat{\boldsymbol{x_{k-1}}} + B_k \boldsymbol{u_k} \tag{7}$$

$$P_k = F_k P_{k-1} F_k^T + Q_k \tag{8}$$

Update

$$\hat{\boldsymbol{x_k}} := \hat{\boldsymbol{x_k}} + K_k(\boldsymbol{z_k} - H_k \hat{\boldsymbol{x_k}}) \tag{9}$$

$$P_k := P_k - K_k H_k P_k \tag{10}$$

$$K_k = P_k H_K^T (H_k P_k H_k^T + R_k)^{-1} \tag{11}$$

- $\boldsymbol{x_k} = \begin{bmatrix} x_k \\ \dot{x_k} \end{bmatrix}$ and is the state of the system
- F: state transition matrix
- $\boldsymbol{u}$: control variable - usually has the acceleration
- B: control matrix - maps control to state variables
- P: state covariance matrix - error of estimates
- Q: process covariance matrix - error due to process
- $\boldsymbol{z}$: measurement from sensors
- H: measurement matrix - maps measurements into state
- K_k: Kalman gain
- R: Measurement covariance matrix - error from measurements

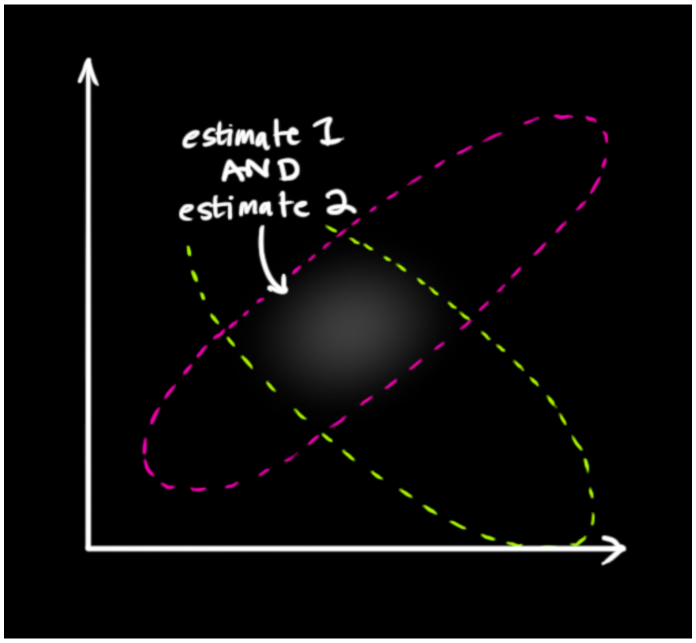

Figure 1: Overlapping Gaussians

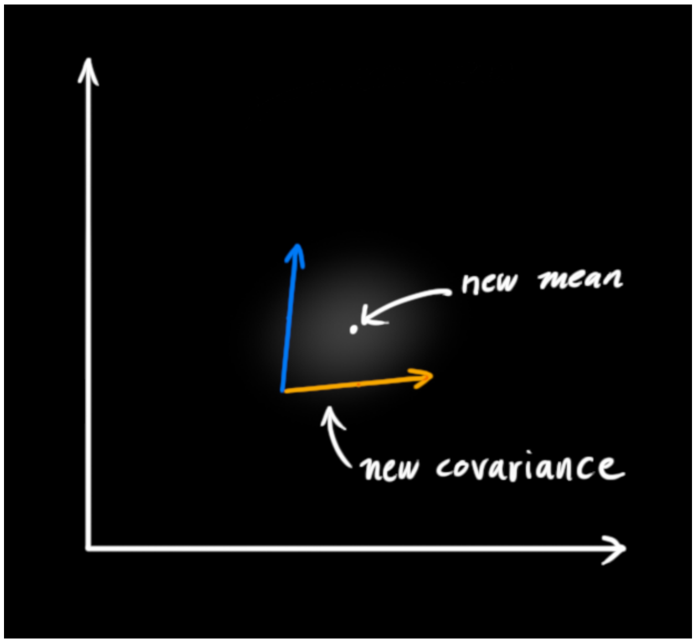

Figure 2: New variance