A Brief Introduction to the Kalman Filter and a Pairs Trading Strategy

October 2, 2021

The Kalman filter is an algorithm which provides an estimate of the "true state" of a physical system by integrating the estimate of the previous state with available observations at the current time. We provide a detailed exposition of what this means via a simple motivating example.

We then describe a method to use the Kalman Filter for a long-short trading strategy.

1 Example of a Simple Physical System

Here we introduce a simple example of a physical system which provides motivation for the algorithm known as the Kalman filter.

The Kalman filter consists of an input u, the state of a system x, a system of equations which tells us how we the state of the system at the next time step $t + \Delta t$ is related to the current state and u, a measurement y, which indirectly gives information about the state of the system x.

Suppose we have a vehicle, vessel, or spaceship whose position we want to track. For example, it is crucial for us to know the exact location of a spaceship so that it arrive to its destination without crashing, and when using a navigation system we don't make the wrong turn.

To simplify the problem, let us consider a car that runs on a straight road for some distance, say 1 km. The car is able to measure its velocity via a speedometer. There is a GPS system that is able to track the position of the car on the road. Neither of these are perfect; there will always be some error in measurement of both velocity and position, although in this particular case, GPS systems tend be quite accurate. At each time step, we record the position reading from the GPS and velocity from the speedometer and use these information to estimate the true position with more certainty than if we were to only use one of those datum.

Denote x as the horizontal displacement of the car from the initial point; denote u as the velocity of the car. Given time t, denote $x(t)$ and $u(t)$ as the displacement and velocity at time t, respectively. Now to be precise, we need another variable which may at this point seem redundant. Define $y = x$. We do

this because the internal "true" state of the system that we want to know is the displacement, which we denote as x, and the measureable quantity that we will use to judge the accuracy of our estimation is displacement itself. In general we might only be able to measure some quantity that is only affected by the true state, so we keep it as a separate variable y.

Given any time t, assume that we have available an estimate of the true value of $x(t)$. Denote this as $\hat{x}(t)$. Then given small enough Δt, we know that the following approximation holds.

$$\hat{x}(t + \Delta t) \approx \hat{x}(t) + u(t)\Delta t \tag{1}$$

$$\hat{y}(t + \Delta t) = \hat{x}(t + \Delta t)$$

If the velocity u does not change throughout the journey, then the equation holds exactly. Because x and u denote the exact theoretical location, and not the measurement location, this is what we shall call our "estimation model". The "true state of the system", which we would like to estimate, is x. Note that given true velocity u, and true position x at time t, then this model holds (within reasonable approximation). However, in practice, we only have an estimate of x at time t, and our measurement of velocity at t. Therefore the best estimation of the next time step with only these two pieces of information is given by equation 1.

Now we know that the measurement of both x and u is noisy. Assume that given a measurement $x(t)$ by the GPS system, the true value of the displacement is distributed by

$$x(t) + v$$

where w is normally distributed $N(0, \sigma_x^2)$. Similarly, assume that given measurement $u(t)$ by the velocimeter, the true value of the velocity is distributed by

$$u(t) + w$$

where v is normally distributed $N(0, \sigma_u^2)$.

The "measurement model" is therefore as follows.

$$\begin{aligned} x(t + \Delta t) &\approx x(t) + (u(t) + w)\Delta t \\ &= x(t) + u(t)\Delta t + w\Delta t \\ y(t + \Delta t) &= x(t + \Delta t) + v \end{aligned} \tag{2}$$

Note that $x(t + \Delta t)$ has a random term due to inaccuracies in the reading of the speedometer, while $y(t + \Delta t)$ has a random term due to the reading error of the GPS system.

We choose to measure the position and velocity N times, and denote $x(t) := x_t$. Then the next increment will be $x_{t+1} := x(t + \Delta t)$. Equation 1 becomes

$$\hat{x}_{t+1} = \hat{x}_t + u_t \Delta t$$

$$\hat{y}_{t+1} = \hat{x}_{t+1}$$

and equation 2 becomes

$$x_{t+1} = x_t + u_t \Delta t + w \Delta t$$

$$y_{t+1} = x_{t+1} + w$$

2 The Algorithm for the Physical System

We initialize four vectors, $\hat{x}, \hat{x}^-, \hat{x}^+, \hat{y}, u$.

At the initial time $t = 0$, we need to specify x and u. We have that u is the input, and is always provided. In our case, $x = y$ and is therefore measureable, but in general, since x is the internal state of the system that can't be directly measured, we wouldn't be able to give a good specification for x. However, as the algorithm progresses, it gives increasingly accurate estimates of x, and this converging property allows us to be reasonably confident of the estimations of x after a few time steps.

So suppose we are given x_t and u_t, where t need not be 0. We would like to find a good estimate of the displacement, which we write as $\hat{x}_t$. Define $\hat{x}^-(t+1)$ by putting

$$\hat{x}^-_{t+1} := \hat{x}_t + u_t \Delta t$$

since we know that if the car was indeed travelling at u_t, then the equation above is a valid estimate for the next true state.

Now if the next state is indeed $\hat{x}^-_{t+1}$, then the measurement which is obtained at the next state should be given by the equation

$$\hat{y} = \hat{x}^-_{t+1}$$

We can now compare the measurement y_t with $\hat{y}_t$. From these two pieces of information, we would be inclined to believe that the true y is between these two values. Denote the new estimate which we take between y_t and $\hat{y}_t$ as $\overline{y}$. Furthermore, if y_t and $\hat{y}_t$ are close together, then it gives us higher confidence in $\overline{y}$, while if they are far apart, we have lower confidence in $\overline{y}$. In fact, if D is the distribution of the measurement y_t (the distribution of the true value given that we measured y_t from the GPS system), and if D' is the distribution of $\hat{y}$ which was obtained from the estimation of the internal state, then the distribution of $\overline{y}$ is given as the (normalized) multiplication of D with D'.

From this more accurate estimation of y_{t+1}, we can work backwards to obtain a more accurate estimation of x_{t+1} than just by using x_t and u_t or $\hat{y}_{t+1}$ alone. Denote this estimation as $\hat{x}^+_{t+1}$. In our problem, this is related by equality, so we have

$$\hat{x}^+_{t+1} := \hat{y}_{t+1}$$

and we set

$$\hat{x}_{t+1} := \hat{x}^+_{t+1}$$

and repeat the calculation for $t+1$ in place of t. What about the distribution of $\hat{x}_{t+1}$? It carries over from $\hat{y}_{t+1}$. However, in general the relation between x and y is more complex.

3 The Algorithm (General Case)

Given time step t, we assume that we are given a vector x_t consisting of n numerical values that characterize the (true) state of a system. We also assume that we are given an (true) input u_t, which relates x_t to x_{t+1} by some equation

$$x_{t+1} = Ax_t + Bu_t + w_t \tag{3}$$

where A and B are $n \times n$ matrices and w_k is a random vector of size n, normally distributed, with covariance matrix Q. In this case, we see that x_{t+1} are linearly related to x_t and u_t. Here w_k accounts for so called "process noise". That is to say, x_t and u_t determine x_{t+1} up to a certain probabilistic distribution. It might not be the case that $x_{t+1} = Ax_t + Bu_t$ holds theoretically.

For example, consider if we wanted to conduct some arbitrary chemical experiment wherein two chambers of gas is to be maintained at the same temperature. These two chambers have thermometers between them. Chamber 1 is poorly insulated and is placed outside of our laboratory while chamber 2 is very well insulated and is inside our laboratory. Thus chamber 1 is exposed to the outside weather, which may be hot or cold. We are able to heat up or cool down chamber 2 at will, but we rely on a system that tries to heat or cool chamber 1 so that it remains the same temperature as chamber 2. When the temperature deviates by more than 3 degrees, it cools chamber 1 by turning on a refrigeration system and heats it by turning on a gas stove located beneath it until the reading from the thermometer is reached. We can then put x_t as the temperature of chamber 1, $A = 1, B = 1$, and u_t as the change in temperature of chamber 2. Now even if we kept the temperature of chamber 2 constant, there will still be variation in chamber 1 because the stove or the compressor either turns on or off and cools or heats the chamber each 10 minutes. Assume that our time step is one hour. Then each hour we don't know whether the chamber has just been cooled or heated or whether it had been left for a while but had not deviated by 3 degrees yet, and therefore there may be noise in the next state.

Alternatively, it is also possible that $x_{t+1} = Ax_t + Bu_t$ holds theoretically as in the case of our car, but the reading of u_t might be noisy.

We also assume that the state x_{t+1} is related to some observable measurement y_{t+1} by a linear equation

$$y_{t+1} = Cx_{t+1} + v_t$$

where if y is of dimension m, we have that C is of dimension $m \times n$, and where v_t is is a random vector of size n, normally distributed, with covariance matrix R. In this case, v_t is called "measurement noise" to mean that even with the

same state x_{t+1}, the repeated measurement takes a probabistic distribution (c.f. quantum mechanics?).

$A.B, C, R, Q$ need to be defined and are in most cases taken to be constant throughout time. Once we have our model and defined these matrices we can start the calculation to obtain the estimated state and the distribution curve of the true state at each time step.

Given the estimated state $\hat{x}_t$, we define

$$\hat{x}_{t+1}^- := Ax_t + Bu_t$$

This is known as the "a-priori state estimate" or the "predicted state estimate". Define

$$P_{t+1}^- = AP_tA^T + Q$$

This matrix is known as the "a-priori state error covariance". In the case that $t = 0$, the matrix P_t can either be the zero matrix (in the case that we are exactly sure of the true state of the system at $t = 0$), or if we know that the true state is distributed like $N(x_0, \pi)$, then it is to be put as π.

Then it is a mathematical fact that given the only the state x_t and the input u_t, the next true state at time $t + 1$ of the system is distributed like $N(\hat{x}_{t+1}^-, P_{t+1}^-)$. We omit the proof.

Now assume that $CP_{t+1}^-C^T + R$ is not degenerate (in practical cases, when we define parameters C and R this will not happen). Define the matrix K_{t+1} as follows.

$$K_{t+1} := P_{t+1}^-C^T \left(CP_{t+1}^-C^T + R\right)^{-1}$$

Now we need the measurement y_t to obtain the best estimate for the next state. The "posteriori state estimate" or the "optimal state estimate" is defined as follows.

$$\hat{x}_{t+1} := \hat{x}_{t+1}^- + K_{t+1}(y_{t+1} - C\hat{x}_{t+1}^-)$$

We will not define in precise terms what the "optimal estimate" is nor will we prove that this in fact gives the optimal estimate of the next state.

Now define

$$P_{t+1} := (I - K_{t+1}C)P_{t+1}^-$$

Then it is a mathematical fact that given the information x_t and the input u_t, along with the measurement y_{t+1} at time $t + 1$, the true state of the system is distributed like $N(\hat{x}_{t+1}, P_{t+1})$. Again, we omit the proof.

Noticing that we gave $\hat{x}_{t+1}$ and P_{t+1}, this information is enough to move to the next step of the algorithm at $t + 2$, where u_{t+1} is given and y_{t+2} is to be measured.

4 Long-Short Strategy

When two (or more) price series have some sort of correlation in between them they typically allow for cointegration into a tradable single prices series given some ratio maintained between them. Assume that two price series follows

some unknown parameterizable stochastic equation which allow cointegration. Then if the parameters of this equation changes due to some changing market condition, then this gives us a problem; our original ratio of the two assets is no longer valid.

Suppose that asset y is related to asset h by some linear equation $y = mh+b$. This is saying that we would be able to predict y by looking at h. This is never exactly the case with any two assets, so we expect some variability when we actually compare the true value of y. We assume that this is Gaussian distributed:

$$y = mh + b + v; \quad v \sim N(0, R)$$

So in particular, if m is unchanging, then it suffices to perform the usual linear regression with, for example, OLS.

We proceed to define the algorithm in for our strategy.

Define the "state" of the system as m. It is possible that m changes around with some process noise. That is, we assume that m follows a random walk.

$$m_{t+1} = \begin{pmatrix} m_t \\ b_t \end{pmatrix} + w_t; \quad w \sim N(0, Q)$$

Note that this specifies a model according to equation 3, where $A = 1$ and there is no input u, or alternatively, we can consider B to be zero. If one knows that oil or gold prices affects the USD/JPY for example, then it may be possible to use it as an input. However since we need to specify its linear relation with m, and it might not be easy to do so.

Our algorithm can be implemented if we specify the matrices $A.B, C, R, Q$. We have that $A = I$, the identity matrix, $B = 0$, $C_t = (h_t, 1)$, a row vector, and R and Q are chosen judiciously by the trader to suit the specific case that he is dealing with.

In particular, one sensibly puts $Q = \alpha I$, where α is a positive small real number.

The following is an example implementation in python3. We assume numpy is imported as "np".

```
# Kalman filter.
# We are given Y as the N by 2 matrix which gives prices of a pair of assets.
# Assume that N is defined.
# Initialize inputs
h = Y[:,0,None]
y = Y[:,1,None]
C = np.append(h,np.ones((N,1)),axis=1)
# Initialize variables to be calculated and recorded.
m = np.zeros((N,2))
P = np.zeros((N,2,2))
# set up initial values of the internal state of the system as a guess.
# In principle this cannot be ‘‘known’’.
# there are infinite solutions for m given y[0,0] and x[0,0],
```

```
# so we chose one such that the intercept is 0.
# the algorithm will find the optimal intercept as time progresses.
m[0,0] = y[0,0]/h[0,0]
m[0,1] = 0
# specify R and Q matrices.
# These are to be changed according to the trading pair.
Q = np.array([[0,0.01],[0,1]]) #  In the case of stock prices, it is usually necessa
# for the variance of the intercept be highly variable.
R = 0.1
# Start the loop
for i in range(N-1):
m_minus = m[i,:,None]
P_minus = P[i,:,:] + Q
Numer = np.dot(P_minus, C[i+1,:,None])
Denom = 1/(np.dot(np.transpose(C[i+1,:,None]), np.dot(P_minus, C[i+1,:,None])) + R)
K = np.dot(Numer, Denom)
Adjust = np.dot(K, y[i+1,:,None] - np.dot(np.transpose(C[i+1,:,None]), m_minus))
m[i+1,:] = np.transpose(m_minus + Adjust)
P[i+1,:,:] = np.dot((np.eye(2,2) - np.dot(K,np.transpose(C[i+1,:,None]))), P_minus)
```