Kalman Filter

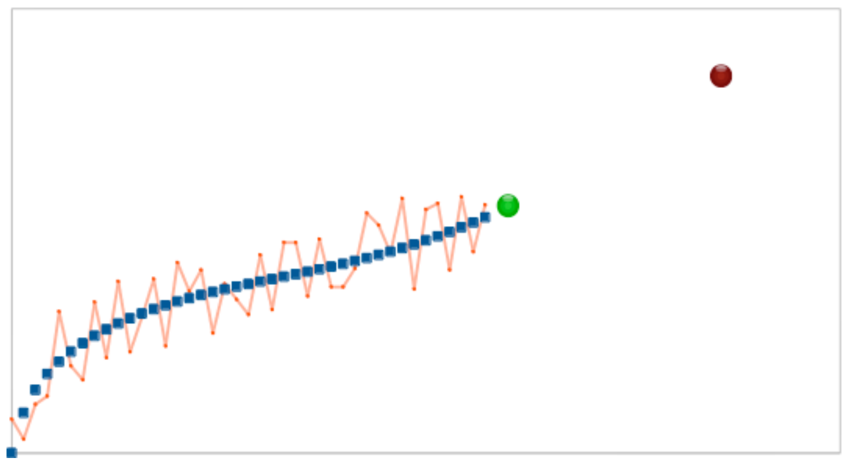
Seoul AI Meetup

Content

- Introduction
- Theory
- Implementation sample
- Discussion

Introduction

- The Kalman filter is an efficient recursive filter that estimates the internal state of a linear dynamic system from a series of noisy measurements.
- It's a method of predicting the future system based on previous ones.



- Given the data in blue, it be reasonable to predict that the green dot should follow, by simply extrapolating the linear trend from the few previous samples. We would be less confidence about predicting dark red point on the right using that method.
- In real world, we don't have the blue line, but the red line ☺

Introduction

- Lessons learned
  - It's not good enough to give a prediction - you also want to know the confidence level.
  - Predicting far ahead into the future is less reliable than nearer predictions.
  - The reliability of your data (the noise), influences the reliability of your predictions.

Theory

- Let's model the prediction in previous example
- First, we need a **state**
  - **State** is a description of all the parameters we will need to describe the current system and perform the prediction
- In this example, we need two numbers – current position and current slope

$$y(t) = y(t-1) + m(t-1)$$

$$m(t) = m(t-1)$$

- Expressing that in form of matrix

$$\mathbf{x}_t = \begin{pmatrix} y(t) \\ m(t) \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} y(t-1) \\ m(t-1) \end{pmatrix} \equiv F\mathbf{x}_{t-1}$$

Theory

- This is a model for blue line
- But we need to model the red line
- So, we add an additional term for **process noise**

$$\mathbf{x}_t = F\mathbf{x}_{t-1} + \mathbf{v}_{t-1}$$

- And what else we need is modeling the measurement
- When we get new data, our parameters should change slightly to refine our current model. Since we only measure y

$$\text{measurement} = \begin{pmatrix} 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} y(t) \\ m(t) \end{pmatrix}$$

- Putting that in matrix, where **w** is measurement noise

$$\mathbf{z}_t = H\mathbf{x}_t + \mathbf{w}_t$$

Theory

- Let's assume our model is perfect, in that case our prediction will be

$$\hat{\mathbf{x}}_{t+1} = F\mathbf{x}_t$$

- What we expect to measure

$$\hat{\mathbf{z}}_{t+1} = H\hat{\mathbf{x}}_{t+1}$$

- Our actual measurement

$$\mathbf{y} \equiv \mathbf{z}_{t+1} - \hat{\mathbf{z}}_{t+1} \neq \mathbf{0}$$

- Here, y is called innovation – it represents how wrong we are
- Using this innovation, we update our previous prediction

$$\hat{\mathbf{x}}_{t+1} = F\mathbf{x}_t + W\mathbf{y}$$

- Here W is called Kalman Gain, and we expect it to be as below

$$W \sim \frac{\text{Process Noise}}{\text{Measurement Noise}}$$

Theory

- How to we evaluate the uncertainty of a value – **variance**!
- Variance of our prediction of state

$$P_t = Cov(\hat{\mathbf{x}}_t)$$

- Like x, P can be derived from previous state

$$P_{t+1} = Cov(\hat{\mathbf{x}}_{t+1}) = Cov(F\mathbf{x}_t) = FCov(\mathbf{x}_t)F^\top = FP_tF^\top$$

- Covariance is also not perfect, so adding covariance matrix of process noise

$$P_{t+1} = FP_tF^\top + Q$$

- Following the same logic

$$S_{t+1} = Cov(\hat{\mathbf{z}}_{t+1}) = Cov(H\hat{\mathbf{x}}_{t+1}) = HCov(\hat{\mathbf{x}}_{t+1})H^\top = HP_{t+1}H^\top$$

$$S_{t+1} = HP_{t+1}H^\top + R$$

Theory

- Finally we can obtain *W* by looking at how two normally distributed states are combined (predicted and measured)

$$W = P_{t+1} H^\top S_{t+1}^{-1}$$

- Now we have basic understanding, let's jump into Kalman Filter page on Wikipedia

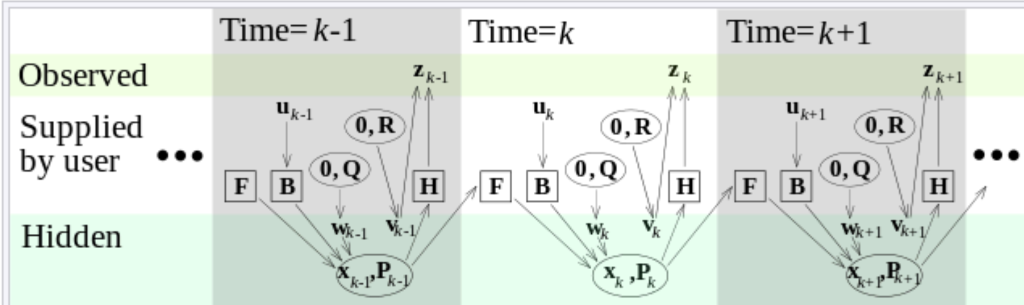
Theory

The Kalman filter model assumes the true state at time k is evolved from the state at $(k-1)$ according to

$$\mathbf{x}_k = \mathbf{F}_k \mathbf{x}_{k-1} + \mathbf{B}_k \mathbf{u}_k + \mathbf{w}_k$$

where

- $\mathbf{F}_k$ is the state transition model which is applied to the previous state $\mathbf{x}_{k-1}$;
- $\mathbf{B}_k$ is the control-input model which is applied to the control vector $\mathbf{u}_k$;
- $\mathbf{w}_k$ is the process noise which is assumed to be drawn from a zero mean multivariate normal distribution, $\mathcal{N}$, with covariance, $\mathbf{Q}_k$: $\mathbf{w}_k \sim \mathcal{N}(0, \mathbf{Q}_k)$.



Model underlying the Kalman filter. Squares represent matrices. Ellipses represent multivariate normal distributions (with the mean and covariance matrix enclosed). Unenclosed values are vectors. In the simple case, the various matrices are constant with time, and thus the subscripts are dropped, but the Kalman filter allows any of them to change each time step.

At time k an observation (or measurement) $\mathbf{z}_k$ of the true state $\mathbf{x}_k$ is made according to

$$\mathbf{z}_k = \mathbf{H}_k \mathbf{x}_k + \mathbf{v}_k$$

where

- $\mathbf{H}_k$ is the observation model which maps the true state space into the observed space and
- $\mathbf{v}_k$ is the observation noise which is assumed to be zero mean Gaussian white noise with covariance $\mathbf{R}_k$: $\mathbf{v}_k \sim \mathcal{N}(0, \mathbf{R}_k)$.

Theory

Predict [edit]

Predicted (*a priori*) state estimate	$\hat{\mathbf{x}}_{k\mid k-1} = \mathbf{F}_k \hat{\mathbf{x}}_{k-1\mid k-1} + \mathbf{B}_k \mathbf{u}_k$
Predicted (*a priori*) estimate covariance	$\mathbf{P}_{k\mid k-1} = \mathbf{F}_k \mathbf{P}_{k-1\mid k-1} \mathbf{F}_k^{\mathrm{T}} + \mathbf{Q}_k$

Update [edit]

Innovation or measurement pre-fit residual	$\tilde{\mathbf{y}}_k = \mathbf{z}_k - \mathbf{H}_k \hat{\mathbf{x}}_{k\mid k-1}$
Innovation (or pre-fit residual) covariance	$\mathbf{S}_k = \mathbf{R}_k + \mathbf{H}_k \mathbf{P}_{k\mid k-1} \mathbf{H}_k^{\mathrm{T}}$
Optimal Kalman gain	$\mathbf{K}_k = \mathbf{P}_{k\mid k-1} \mathbf{H}_k^{\mathrm{T}} \mathbf{S}_k^{-1}$
Updated (*a posteriori*) state estimate	$\hat{\mathbf{x}}_{k\mid k} = \hat{\mathbf{x}}_{k\mid k-1} + \mathbf{K}_k \tilde{\mathbf{y}}_k$
Updated (*a posteriori*) estimate covariance	$\mathbf{P}_{k\mid k} = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k) \mathbf{P}_{k\mid k-1}$
Measurement post-fit residual	$\tilde{\mathbf{y}}_{k\mid k} = \mathbf{z}_k - \mathbf{H}_k \hat{\mathbf{x}}_{k\mid k}$

Implementation

- Let's first take a look at simple source code to that implements Kalman Filter