TIME SERIES 501

Lesson 6: Kalman Filters

Learning Objectives

You will be able to do the following:

- Define “Kalman filter.”
- Explain conceptually how a Kalman filter works.
- Describe the assumptions underlying the Kalman filter approach.
- Use Python* to apply a Kalman filter to time-series data.

What Is a Kalman Filter?

A way to make a best guess about a signal in the presence of noise.

- Fundamental idea: noisy signal is made less noisy (filter out some of the noise)
- Best guess = lowest error
- Applications:
  - tracking objects (Apollo project, GPS, self-driving cars)
  - image processing
  - economic and financial modeling

The Kalman Filter: The Basics

What are the ingredients?

- A mathematical model of the system
- Measurements related to the system
- Example:
  - Car moving at a constant velocity
  - Model is position = velocity*time + system noise
  - Measurements of position and velocity (which are also noisy)

Image: https://www.pexels.com/photo/action-asphalt-auto-automobile-210019/

The Kalman Filter: The Basics

What is the goal?

- Estimate the variables of interest at any given time—your “state.” (In the previous example, these variables are the position and velocity of the car.)
- At each time point, the Kalman filter estimates your state by calculating the optimal compromise between two sources of data:
  - the model with information from the previous time point
  - the latest measurement

The Kalman Filter: The Basics

Where does the noise come in?

- The compromise is determined by the noise.
- If the model has relatively large errors, more weight is given to the latest measurement in making the current estimate.
- If the measurement has larger errors, more weight is given to the model in making the current estimate.
- Therefore, you need to estimate not only your state but also the errors (the covariance) for both the model and the measurements. These must be updated at each time step, too.

Kalman Filter Is a Recursive Algorithm

Helps with practical implementation

- Uses information in previous time step to update estimates.
- Does not require information from all data ever acquired to be kept in memory.
- Has predictor-corrector structure—predict, measure, update, repeat.

Image: https://www.pexels.com/photo/road-car-fast-speed-20411/

Fundamental Assumptions

The Kalman filter approach is based on three fundamental assumptions:

- The system can be described or approximated by a linear model.
- All noise (from both the system and the measurements) is white.
- All noise is Gaussian.

Assumption 1: Linear Model

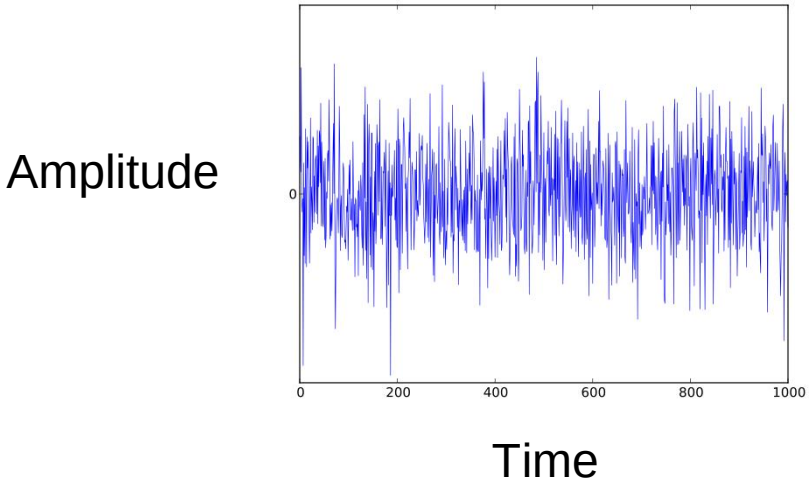
Variable at current time is a linear function of variable and noise at previous time.

- Many systems can be approximated this way.
- Linear systems are more easily analyzed mathematically.
- Nonlinear systems can often be approximated by linear models around a current estimate (for example, extended Kalman filter).

Assumption 2: Whiteness

Noise value is not correlated in time.

- If you know noise now, doesn't help you predict noise at other times.
- Useful, approximate description of real noise.
- Makes mathematics tractable.

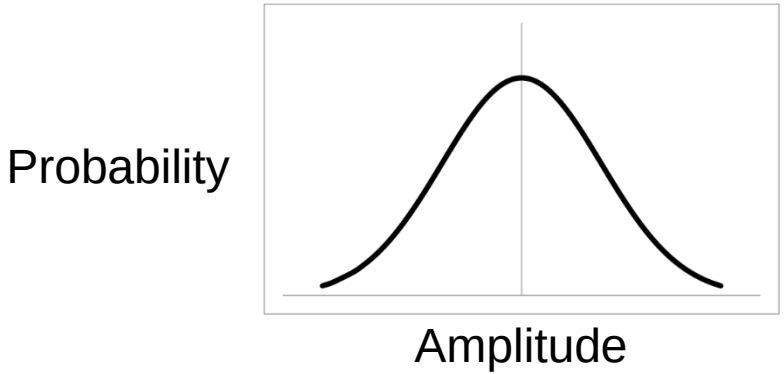


https://commons.wikimedia.org/w/index.php?curid=24084756

Assumption 3: Gaussian Noise

At any point in time, probability density of noise amplitude is a Gaussian.

- System and measurement noise often a combination of many small sources of noise. Can show that net effect is approximately Gaussian.
- If only mean and variance are known (typical case in engineering systems), Gaussian distribution is a good choice as these two quantities completely determine the Gaussian distribution.
- Makes mathematics tractable.



EXAMPLE 1: STATIC SYSTEM

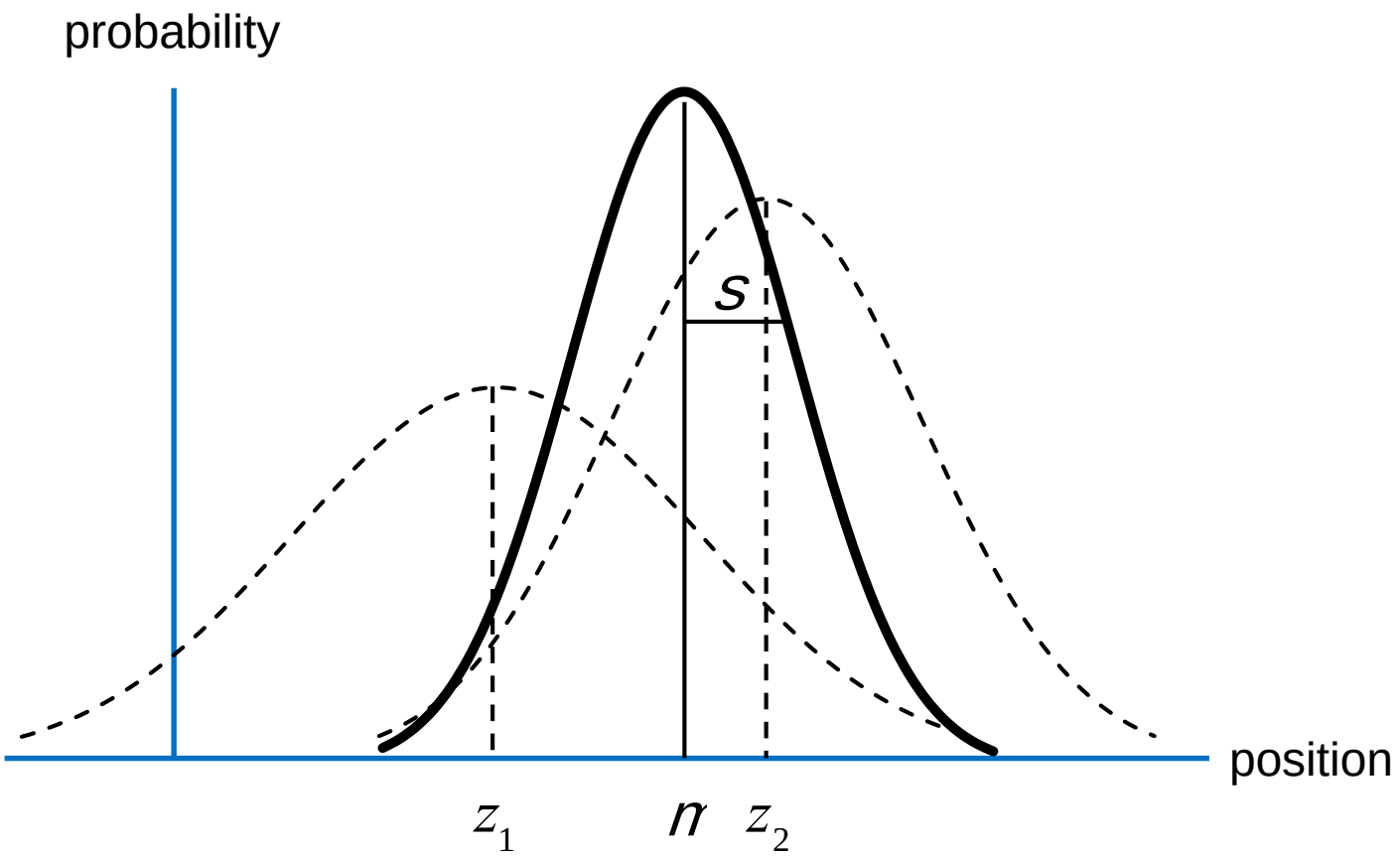
Example: One-Dimensional Position Estimation

You and a friend are lost in car at night on a desolate country road.
You stop to figure out where you are.

- You use stars to determine the position.

  At time t_1, your measurement is z_1 with standard deviation σ_{z_1}
- Your friend uses GPS to determine the position.

  At time t_2 ($\approx t_1$), hers is z_2 with standard deviation σ_{z_2}
- What is the best estimate of your position at the two times?


probability
s
position
z_1
n
z_2

Best Estimate at Time t_1

Estimate of position x at time t_1

$$\hat{x}(t_1) = z_1$$

Variance of the error in the estimate

$$s_x^2(t_1) = s_{z_1}^2$$

Best Estimate at Time t_2

Estimate of position *x* at time t_2

$$\hat{x}(t_2) = m$$

Variance of the error in the estimate

$$s_x^2(t_2) = s^2$$

Best Estimate at Time t_2

$$m = \frac{s_{z_2}^2}{s_{z_1}^2 + s_{z_2}^2} z_1 + \frac{s_{z_1}^2}{s_{z_1}^2 + s_{z_2}^2} z_2$$

$$\frac{1}{s^2} = \frac{1}{s_{z_1}^2} + \frac{1}{s_{z_2}^2}$$

Rewrite Results at Time t_2

$$\hat{x}(t_2) = \hat{x}(t_1) + K(t_2)\left[z_2 - \hat{x}(t_1) \right]$$

$$s_x^2(t_2) = s_x^2(t_1) - K(t_2)s_x^2(t_1)$$

$$K(t_2) = \frac{s_{z_1}^2}{s_{z_1}^2 + s_{z_2}^2}$$

Kalman gain

EXAMPLE 2: DYNAMIC SYSTEM

Add Motion of Car to Example 1

- Car moves at constant speed u from time t_2 to time t_3
- Use information from time t_2 to predict position at time t_3 (before new measurement).
- Measure position at time t_3
- Use measurement to update prediction (using Kalman gain).

Have two key steps—prediction and update.

What Is the Model for the System?

- Previously, car was stationary. Model was x = constant + noise
- Now car is moving with constant velocity *u*, so the model changes.

$$\frac{dx}{dt} = u + w$$

- w represents the noise: uncertainty in value of velocity, small accelerations, etc.
- Modeled as Gaussian white noise with mean zero and variance s_w^2

Prediction of Position Just before Measurement

Car moves at constant speed u from time t_2 to time t_3

- Just before the new measurement at time t_3, the predicted position and error are

$$\hat{x}(t_3^-) = \hat{x}(t_2) + u[t_3^- - t_2]$$

$$s_x^2(t_3^-) = s_x^2(t_2) + s_w^2[t_3^- - t_2]$$

To emphasize that values are predicted before the measurement, denote time as t_3^-

Update Estimated Position after Measurement Is Made

At time t_3 a measurement is made.

- Its value is z_3 with standard deviation σ_{z_3}
- Combine this new information with the prediction made before measurement using the same process as done previously (for the static example).

Update at Time t_3

$$\hat{x}(t_3) = \hat{x}(t_3^-) + K(t_3)\left[z_3 - \hat{x}(t_3^-) \right]$$

$$s_x^2(t_3) = s_x^2(t_3^-) - K(t_3)s_x^2(t_3^-)$$

$$K(t_3) = \frac{s_x^2(t_3^-)}{s_x^2(t_3^-) + s_{z_3}^2}$$

Kalman gain

KALMAN FILTER EQUATIONS

Kalman Filter: General Case

Generalize from 1D dynamic case.

- State is described by a vector.
- Error is described by a covariance matrix.
- Have prediction and update steps as before.
- Drop time for clarity; use subscript to denote time steps.

Kalman Filter: Prediction Step

$$\hat{x}_k = F_k \hat{x}_{k-1} + B_k u_k$$

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

$\hat{x}_k$ state vector estimate

P_k covariance matrix of state vector

F_k prediction matrix

B_k control matrix

u_k control vector

Q_k covariance matrix of model noise

Kalman Filter: Update Step

$$\hat{x}_k^* = \hat{x}_k + K_k^*[z_k - H_k\hat{x}_k]$$

$$P_k^* = P_k - K_k^* H_k P_k$$

z_k measurement vector

H_k measurement matrix

K_k^* Kalman gain matrix

$*$ denotes value after measurement

Kalman Filter: Kalman Gain Matrix

$$K_k^* = P_k H_k^T (H_k P_k H_k^T + R_k)^{-1}$$

R_k covariance matrix of measurement noise

Kalman Filter: The Workflow

1. Make initial estimates of state vector and covariance matrix (time k=0).
2. Predict state and covariance for next time step.
3. Compute the Kalman gain.
4. Make a measurement.
5. Update estimates of state and covariance.
6. Repeat steps 2-5.

APPLICATIONS IN PYTHON

Use Python to Identify and Transform Nonstationarity

Next up is a look at applying these concepts in Python.

- See notebook entitled *Introduction_to_Kalman_Filter_student.ipynb*

Learning Objectives Recap

In this session you learned how to do the following:

- Define “Kalman filter.”
- Describe the assumptions underlying the Kalman filter approach.
- Explain conceptually how a Kalman filter works.
- Use Python to apply a Kalman filter to time-series data.

References

- Stochastic Models, Estimation, and Control (Vol. 1) by Peter S. Maybeck
- http://www.bzarg.com/p/how-a-kalman-filter-works-in-pictures/
- http://www.cl.cam.ac.uk/~rmf25/papers/Understanding the Basis of the Kalman Filter.pdf

Legal Notices and Disclaimers

This presentation is for informational purposes only. INTEL MAKES NO WARRANTIES, EXPRESS OR IMPLIED, IN THIS SUMMARY.

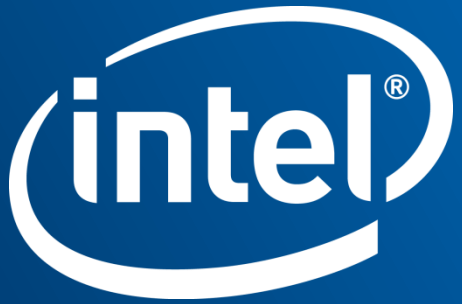
Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Performance varies depending on system configuration. Check with your system manufacturer or retailer or learn more at intel.com.

Sample source code is released under the Intel Sample Source Code License Agreement.

Intel, the Intel logo, the Intel. Experience What's Inside logo, and Intel. Experience What's Inside are trademarks of Intel Corporation in the U.S. and/or other countries.

*Other names and brands may be claimed as the property of others.

Copyright © 2018, Intel Corporation. All rights reserved.


intel®