ACM 116: The Kalman filter

• Example
• General Setup
• Derivation
• Numerical examples
  – Estimating the voltage
  – 1D tracking
  – 2D tracking
Example: Navigation Problem

- Truck on “frictionless” straight rails
- Initial position $X_0 = 0$
- Movement is buffeted by random accelerations
- We measure the position every $\Delta t$ seconds
- State variables $(X_k, V_k)$ position and velocity at time $k\Delta t$

\[
X_k = X_{k-1} + V_{k-1}\Delta t + a_{k-1}\Delta t^2/2 \\
V_k = V_{k-1} + a_{k-1}\Delta t
\]

where $a_{k-1}$ is a random acceleration

- Observations

\[
Y_k = X_k + Z_k
\]

where $Z_k$ is a noise term.

The goal is to estimate the position and velocity at all times.
General Setup

Estimation of a stochastic dynamic system

- Dynamics
  \[ X_k = F_{k-1}X_{k-1} + B_{k-1}u_{k-1} + W_{k-1} \]
  - \( X_k \): state of the system at time \( k \)
  - \( u_{k-1} \): control-input
  - \( W_{k-1} \): noise

- Observations
  \[ Y_k = H_kX_k + Z_k \]
  - \( Y_k \): observed
  - \( Z_k \): noise

- The noise realizations are all independent

- Goal: predict state \( X_k \) from past data \( Y_0, Y_1, \ldots, Y_{k-1} \).
Derivation

Derivation in the simpler model where the dynamics is of the form

\[ X_k = a_{k-1}X_{k-1} + W_{k-1} \]

and the observations

\[ Y_k = X_k + Z_k \]

The objective is to find, for each time \( k \), the minimum MSE filter based on \( Y_0, Y_1, \ldots, Y_{k-1} \)

\[ \hat{X}_k = \sum_{j=1}^{k} h_{j}^{(k-1)} Y_{k-j} \]

To find the filter, we apply the orthogonality principle

\[ E((X_k - \sum_{j=1}^{k} h_{j}^{(k-1)} Y_{k-j})Y_\ell) = 0, \quad \ell = 0, 1, \ldots, k - 1. \]
Recursion

The beautiful thing about the Kalman filter is that one can almost deduce the optimal filter to predict $X_{k+1}$ from that predicting $X_k$.

$$h_{j+1}^{(k)} = (a_k - h_1^{(k)})h_j^{(k-1)}, \quad j = 1, \ldots, k.$$ 

Given the filter $h^{(k-1)}$, we only need to find $h_1^{(k)}$ to get the filter at the next time step.
How to find $h_1^{(k)}$?

Observe that the next prediction is equal to

$$
\hat{X}_{k+1} = h_1^{(k)} Y_k + \sum_{j=1}^{k} (a_k - h_1^{(k)}) h_j^{(k-1)} Y_{k-j}
$$

$$
= a_k \hat{X}_k + h_1^{(k)} (Y_k - \hat{X}_k)
$$

Interpretation

$$
\hat{X}_{k+1} = a_k \hat{X}_k + h_1^{(k)} I_k
$$

- $a_k \hat{X}_k$ is the prediction based on the estimate at time $k$
- $h_1^{(k)} I_k$ is a corrective term which is available since we now see $Y_k$
  - $h_1^{(k)}$ is called the gain
  - $I_k = Y_k - \hat{X}_k$ is called the innovation
Error of Prediction

To find $h_1^{(k)}$, we look at the error of prediction

$$
\epsilon_k = X_k - \hat{X}_k
$$

We have the recursion

$$
\epsilon_{k+1} = (a_k - h_1^{(k)})\epsilon_k + W_k - h_1^{(k)} Z_k
$$

- $\epsilon_0 = Z_0$
- $E(\epsilon_k) = 0$
- $E(\epsilon_k^2) = [a_k - h_1^{(k)}]^2 E(\epsilon_k^2) + E(W_k^2) + [h_1^{(k)}]^2 E(Z_k^2)$
To minimize the MSE $\epsilon_{k+1}$, we adjust $h_1^{(k)}$ so that

$$\partial_{h_1^{(k)}} E(\epsilon_{k+1}^2) = 0 = -2(a_k - h_1^{(k)}) E(\epsilon_k^2) + 2h_1^{(k)} E(Z_k^2)$$

which is given by

$$h_1^{(k)} = \frac{a_k E(\epsilon_k^2)}{E(\epsilon_k^2) + E(Z_k^2)}$$

Note that this gives the recurrence relation

$$E(\epsilon_{k+1}^2) = a_k(a_k - h_1^{(k)}) E(\epsilon_k^2) + E(W_k^2)$$
The Kalman Filter Algorithm

- Initialization $\hat{X}_0 = 0$, $E(\epsilon_0^2) = E(Z_0^2)$

- Loop: for $k = 0, 1, \ldots$

$$h_1^{(k)} = \frac{a_k E(\epsilon_k^2)}{E(\epsilon_k^2) + E(Z_k^2)}$$

$$\hat{X}_{k+1} = a_k \hat{X}_k + h_1^{(k)}(Y_k - \hat{X}_k)$$

$$E(\epsilon_{k+1}^2) = a_k(a_k - h_1^{(k)})E(\epsilon_k^2) + E(W_k^2)$$
Benefits

• Requires no knowledge about the structure of $W_k$ and $Z_k$ (only variances)

• Easy implementation

• Many applications
  – Inertial guidance system
  – Autopilot
  – Satellite navigation system
  – Many others
General Formulation

\[ X_k = F_{k-1}X_{k-1} + W_{k-1} \]
\[ Y_k = H_k X_k + Z_k \]

The covariance of \( W_k \) is \( Q_k \) and that of \( Z_k \) is \( R_k \).

Two variables:

- \( \hat{X}_{k|k} \) estimate of the state at time \( k \) based upon \( Y_0, \ldots, Y_{k-1} \)
- \( E_{k|k} \) error covariance matrix, \( E_{k|k} = \text{Cov}(X_k - \hat{X}_{k|k}) \)
Prediction

\[
\hat{X}_{k+1|k} = F_k \hat{X}_{k|k-1}
\]

\[
E_{k+1|k} = F_k E_{k|k-1} F_k^T + Q_k
\]

Update

\[
I_k = Y_k - H_k \hat{X}_{k+1|k}
\]

Innovation

\[
S_k = H_k E_{k+1|k} H_k^T + R_k
\]

Innovation covariance

\[
K_k = E_{k+1|k} H_k^T S_k^{-1}
\]

Kalman Gain

\[
\hat{X}_{k+1|k+1} = \hat{X}_{k+1|k} + K_k I_k
\]

Updated state estimate

\[
E_{k+1|k+1} = (I_d - K_k H_k) E_{k+1|k}
\]

Updated error covariance
Estimating Constant Voltage

We wish to estimate some voltage which is almost constant except for some small random fluctuations. Our measuring device is imperfect (e.g. because of a poor A/D conversion). The process is governed by:

\[ X_k = X_0 + W_k, \quad k = 1, 2, \ldots \]

with \( X_0 = 0.5V \), and the measurements are

\[ Z_k = X_k + V_k, \quad k = 1, 2, \ldots \]

where \( W_k, V_k \) are uncorrelated Gaussian white noise processes, with \( R := \text{Var}(V_k) = 0.01, \text{Var}(W_k) = 10^{-5}. \)
Accurate knowledge of measurement variance, $R_{est} = R = 0.01$
Optimistic estimate of measurement variance, $R_{est}=0.0001$

Plot showing iteration versus voltage estimate and exact process measurements.
Pessimistic estimate of measurement variance, $R_{est}=1$

Voltage estimate vs. exact process measurements.

- Dashed line: estimate
- Solid line: exact process
- Crosses: measurements
1D Tracking

Estimation of the position of a vehicle.

Let $X$ be a state variable (position and speed), and $A$ is a transition matrix

\[ A = \begin{pmatrix} 1 & \Delta t \\ 0 & 1 \end{pmatrix}. \]

The process is governed by:

\[ X_{n+1} = AX_n + W_n \]

where $W_n$ is a zero-mean Gaussian white noise process. The observation is

\[ Y_n = CX_n + Z_n \]

where the matrix $C$ only picks up the position and $Z_n$ is another zero-mean Gaussian white noise process independent of $W_n$. 
Estimation of a moving vehicle in 1-D.
2D Example

General setup

\[ X(t + 1) = FX(t) + W(t), \quad W \sim N(0, Q), \]
\[ Y(t) = HX(t) + V(t), \quad V \sim N(0, R) \]

Moving particle at constant velocity subject to random perturbations in its trajectory. The new position \((x_1, x_2)\) is the old position plus the velocity \((dx_1, dx_2)\) plus noise \(w\).

\[
\begin{pmatrix}
  x_1(t) \\
  x_2(t) \\
  dx_1(t) \\
  dx_2(t)
\end{pmatrix} =
\begin{pmatrix}
  1 & 0 & 1 & 0 \\
  0 & 1 & 0 & 1 \\
  0 & 0 & 1 & 0 \\
  0 & 0 & 0 & 1
\end{pmatrix}
\begin{pmatrix}
  x_1(t - 1) \\
  x_2(t - 1) \\
  dx_1(t - 1) \\
  dx_2(t - 1)
\end{pmatrix} +
\begin{pmatrix}
  w_1(t - 1) \\
  w_2(t - 1) \\
  dw_1(t - 1) \\
  dw_2(t - 1)
\end{pmatrix}
\]
Observations

We only observe the position of the particle.

\[
\begin{pmatrix}
y_1(t) \\
y_2(t)
\end{pmatrix} = \begin{pmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0
\end{pmatrix} \begin{pmatrix}
x_1(t) \\
x_2(t) \\
dx_1(t) \\
dx_2(t)
\end{pmatrix} + \begin{pmatrix}
v_1(t) \\
v_2(t)
\end{pmatrix}
\]

Source: http://www.cs.ubc.ca/~murphyk/Software/Kalman/kalman.html
Implementation

% Make a point move in the 2D plane
% State = (x y xdot ydot). We only observe (x y).

% This code was used to generate Figure 17.9 of
% "Artificial Intelligence: a Modern Approach",

% X(t+1) = F X(t) + noise(Q)
% Y(t) = H X(t) + noise(R)

ss = 4; % state size
os = 2; % observation size
F = [1 0 1 0; 0 1 0 1; 0 0 1 0; 0 0 0 1];
H = [1 0 0 0; 0 1 0 0];
Q = 1*eye(ss);
R = 10*eye(os);
initx = [10 10 1 0]';
initV = 10*eye(ss);
seed = 8; rand('state', seed);
randn('state', seed);
T = 50;
[x,y] = sample_lds(F,H,Q,R,initx,T);
Apply Kalman Filter

```matlab
[xfilt, Vfilt] = kalman_filter(y, F, H, Q, R, initx, initV);
dfilt = x([1 2], :) - xfilt([1 2], :);
mse_filt = sqrt(sum(sum(dfilt.^2)))
figure;
plot(x(1,:), x(2,:), 'ks-');
hold on
plot(y(1,:), y(2,:), 'g*');
plot(xfilt(1,:), xfilt(2,:), 'rx:');
hold off
legend('true', 'observed', 'filtered', 0)
xlabel('X1'), ylabel('X2')
```
Apply Kalman Smoother

[xsmooth, Vsmooth] = kalman_smoother(y,F,H,Q,R,initx,initV);
dsmooth = x([1 2],:) - xsmooth([1 2],:);
mse_smooth = sqrt(sum(sum(dsmooth.^2)))

figure;
hold on
plot(x(1,:), x(2,:), 'ks-');
plot(y(1,:), y(2,:), 'g*');
plot(xsmooth(1,:), xsmooth(2,:), 'rx:');
hold off
legend('true', 'observed', 'smoothed', 0)
xlabel('X1'), ylabel('X2')