Kalman filter and smoother

Contents

The Kalman filter is a method of estimating the current state of a dynamical system, given the observations so far. The underlying model is a hidden Markov model (HMM) in which everything is multivariate normal—so in particular, the hidden variables are continuous, rather than discrete. The Kalman filter is actually just the forward algorithm, except that each step can be computed analytically due to the magic of Gaussians. As one might expect, there is also a backward algorithm (or something very similar), and this is referred to as the smoother algorithm. The smoother allows one to refine estimates of previous states, in the light of later observations. As in the case of discrete-state HMMs, the results of the Kalman filter and smoother can also be combined with expectation-maximization to estimate the parameters of the model. I think it is fair to say that the Kalman filter is one of the most important algorithms of the 20th century.

1 Background

- Around 1960, the United States and the Soviet Union were developing rocket technology for their space programs (and, no coincidence, for intercontinental ballistic missiles as well).
- At the same time, Rudolph Kalman in the US and Ruslan Stratonovich in the USSR were developing methods for efficiently and accurately estimating the state of a dynamical system by accumulating noisy measurements from many different instruments over time. Kalman's method would later become known as the Kalman filter, and is a special case of Stratonovich's method. Early contributions were also made by Thorvald Thiele, Peter Swerling, and Richard Bucy.
- When Kalman visited NASA Ames Research Center, Stanley Schmidt realized the utility of Kalman's work for navigation and control of aircraft and spacecraft, and the Kalman filter became an integral part of the Apollo navigation computer.
- Today, essentially all high-performance navigation systems use a Kalman filter or some variant thereof. The method is used in rockets, missiles, spacecraft including the international space station, unmanned aerial vehicles, ground robots, and recently, self-driving cars.

2 Model

- The Kalman filter and smoother are based on the following probabilistic model.
- Like a discrete-state HMM, the sequence of observations $x_1, \ldots, x_n$ is modeled jointly along with a sequence of hidden states $z_1, \ldots, z_n$ by a distribution that respects the graph:

$$\begin{tikzcd}
Z_1 \arrow[r] \arrow[d] & Z_2 \arrow[r] \arrow[d] & Z_3 \arrow[r] \arrow[d] & \cdots \arrow[r] & Z_n \arrow[d] \\
X_1 & X_2 & X_3 & \cdots & X_n
\end{tikzcd}$$

  In other words, we assume

$$p(x_{1:n}, z_{1:n}) = p(z_1)p(x_1|z_1)\prod_{j=2}^{n} p(z_j|z_{j-1})p(x_j|z_j).$$

- However, unlike a discrete-state HMM, each hidden state z_j is modeled as a continuous random variable in $\mathbb{R}^d$ with a multivariate normal distribution.

- Specifically, the initial distribution $p(z_1)$, the transition distributions $p(z_j|z_{j-1})$ (a.k.a. the "process model"), and the emission distributions $p(x_j|z_j)$ (a.k.a. the "measurement model") are assumed to be

$$\begin{aligned}p(z_1) &= \mathcal{N}(z_1 \mid \mu_0, V_0)\\ p(z_j|z_{j-1}) &= \mathcal{N}(z_j \mid Fz_{j-1}, Q)\\ p(x_j|z_j) &= \mathcal{N}(x_j \mid Hz_j, R)\end{aligned}$$

  where

  - $z_j \in \mathbb{R}^d$ (the state of the system at time step j),
  - $x_j \in \mathbb{R}^D$ (the measurements at time step j),
  - $\mu_0 \in \mathbb{R}^d$ is an arbitrary vector (the initial mean, our "best guess" at the initial state),
  - $V_0 \in \mathbb{R}^{d\times d}$ is a symmetric positive definite matrix (the initial covariance matrix, quantifying our uncertainty about the initial state),
  - $F \in \mathbb{R}^{d\times d}$ is an arbitrary matrix (modeling the physics of the process, or a linear approximation thereof),
  - $Q \in \mathbb{R}^{d\times d}$ is a symmetric positive definite matrix (quantifying the noise/error in the process that is not captured by F),
  - $H \in \mathbb{R}^{D\times d}$ is an arbitrary matrix (relating the measurements to the state),
  - $R \in \mathbb{R}^{D\times D}$ is a symmetric positive definite matrix (quantifying the noise/error of the measurements).

- The model can easily be extended to handle time-dependence in F, Q, H, and R, by simply replacing them with F_j, Q_j, H_j, and R_j in the expressions above. The Kalman filter and smoother algorithms can easily be modified to handle this generalization.

2.1 Example

- To illustrate with a simple example, consider an object that has been launched into the air. For simplicity, let's only consider the vertical dimension, height. Suppose the hidden state z_j consists of the acceleration a_j, velocity v_j, and position r_j at time t_j, specifically,

$$z_j = \begin{bmatrix} a_j \\ v_j \\ r_j \end{bmatrix}.$$

- From basic physics, we know that acceleration is the time-derivative of velocity, and velocity is the time-derivative of position:

$$\begin{aligned}a(t) &= \frac{d}{dt}v(t)\\ v(t) &= \frac{d}{dt}r(t).\end{aligned}$$

- Therefore, by first-order Taylor approximations to $v_{j+1} = v(t_{j+1})$ and $r_{j+1} = r(t_{j+1})$,

$$\begin{aligned} v_{j+1} &\approx v_j + (t_{j+1} - t_j)a_j \\ r_{j+1} &\approx r_j + (t_{j+1} - t_j)v_j. \end{aligned}$$

  These approximations will be good when $t_{j+1} - t_j$ is small. (A slight improvement for r_{j+1} could easily be obtained by using a second-order Taylor approximation, leading to $r_{j+1} \approx r_j + (t_{j+1} - t_j)v_j + \frac{1}{2}(t_{j+1} - t_j)^2 a_j$.)

- This suggests a way of predicting the velocity and position at time step $j + 1$ given the state at time step j. How could we predict the acceleration, a_{j+1}? The simplest possible thing would be to assume that it is close to a_j, in other words, $a_{j+1} \approx a_j$. (A more sophisticated approach would be to also account for air resistance, i.e., drag, and any other known forces.)

- For simplicity, let's assume $dt = t_{j+1} - t_j$ is the same for all j. Then these approximations lead us to choose:

$$F = \begin{bmatrix} 1 & 0 & 0 \\ dt & 1 & 0 \\ 0 & dt & 1 \end{bmatrix},$$

  so that in the model, the transition distribution $p(z_{j+1}|z_j) = \mathcal{N}(z_{j+1} \mid Fz_j, Q)$ has mean

$$Fz_j = \begin{bmatrix} a_j \\ v_j + a_j dt \\ r_j + v_j dt \end{bmatrix}.$$

- How should we choose H? Suppose that at each time step j, we observe a noisy measurement of acceleration, $\tilde{a}_j$, and a noisy measurement of position, $\tilde{r}_j$:

$$x_j = \begin{bmatrix} \tilde{a}_j \\ \tilde{r}_j \end{bmatrix}.$$

  If we assume the measurements are unbiased, then this implies

$$H = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

  so that in the model, the emission distribution $p(x_j|z_j) = \mathcal{N}(x_j \mid Hz_j, R)$ has mean

$$Hz_j = \begin{bmatrix} a_j \\ r_j \end{bmatrix}.$$

- The measurement covariance matrix R should be chosen based on the noise/error of the instruments used to measure acceleration and position, for example, something like

$$R = \begin{bmatrix} \sigma_a^2 & 0 \\ 0 & \sigma_r^2 \end{bmatrix},$$

  assuming the errors are uncorrelated. Appropriate choices of σ_a^2 and σ_r^2 can be obtained from manufacturer's specifications and/or from calibration experiments, or they could be estimated from the data itself.

- How should we choose Q? This is a bit trickier since the accuracy of the physical model might not be obvious, *a priori*. One approach is to estimate Q based on the data. We won't go into details about how to do this, but a simple way is to first choose an initial guess for Q, run the Kalman filter and smoother algorithms to estimate the trajectory, and then estimate Q based on this inferred trajectory. To make an initial guess of Q with roughly the right order of magnitude, here is one possibility: if we expect the approximation error to accumulate like Brownian motion, then the error after dt time should have variance of order dt. A more principled approach would be to use expectation-maximization.

- As usual, it's a good idea to do a sensitivity analysis to see how much the results depend on the choice of parameters, especially for parameters about which you are most uncertain.

3 Overview of inference algorithms

- As in the case of a discrete-state HMM, there is a forward algorithm (the Kalman filter) and a backward algorithm (the Rauch–Tung–Striebel smoother), except that now, each step involves an integral instead of a sum. Due to the linear-Gaussian assumptions in the model, these integrals can be computed analytically, leading to algorithms that are mathematically elegant and computationally efficient.

- Recall that for a discrete-state HMM, the forward and backward algorithms involve computing $s_j(z_j) = p(x_{1:j}, z_j)$ and $r_j(z_j) = p(x_{j+1:n}|z_j)$ for every possible value of z_j. In a continuous-state model, however, we cannot compute these quantities for every possible value of z_j, since there are infinitely many possible values.

- What we can do, instead, is to work in terms of a parametric representation. This involves a slight change of perspective. First, note that

$$p(z_j|x_{1:j}) \propto s_j(z_j), \text{ and}$$
$$p(z_j|x_{1:n}) \propto s_j(z_j)r_j(z_j).$$

  It turns out that under the assumed model, these are multivariate normal distributions, in other words,

$$p(z_j|x_{1:j}) = \mathcal{N}(z_j|\mu_j, V_j)$$
$$p(z_j|x_{1:n}) = \mathcal{N}(z_j|\hat{\mu}_j, \hat{V}_j)$$

  for some $\mu_j, V_j, \hat{\mu}_j, \hat{V}_j$. Consequently, we can reformulate the forward and backward algorithms in terms of the parameters $\mu_j, V_j, \hat{\mu}_j, \hat{V}_j$, rather than computing the density at every point.

- The Kalman filter is identical to the forward algorithm for discrete-state HMMs, except that it is expressed in terms of μ_j and V_j instead of $s_j(z_j)$ (and the derivation involves an integral instead of a sum). So, even though the derivation of the Kalman filter may seem a bit complicated, remember that all we are doing is expressing the forward algorithm in terms of multivariate normal parameters.

- The Rauch–Tung–Striebel (RTS) smoother is similar to (but not identical to) the backward algorithm, and is expressed in terms of $\hat{\mu}_j$ and $\hat{V}_j$ rather than $r_j(z_j)$. (The main difference between the RTS smoother and the backward algorithm is that the RTS smoother works in terms of $s_j(z_j)r_j(z_j)$ instead of $r_j(z_j)$.)

3.1 Two very useful properties of multivariate normals

- Multivariate normal distributions have a couple of special properties that we will use over and over again the derivation of the Kalman filter and smoother.

  (a) $\displaystyle\int \mathcal{N}(y|Az+b,C)\mathcal{N}(z|m,V)dz = \mathcal{N}(y \mid Am+b, AVA^{\mathsf{T}}+C)$,

  (b) $\mathcal{N}(y|Az+b,C)\mathcal{N}(z|m,V) \underset{z}{\propto} \mathcal{N}\big(z \,\big|\, m+K(y-(Am+b)), (I-KA)V\big)$

  where
  $$K = VA^{\mathsf{T}}(AVA^{\mathsf{T}}+C)^{-1}.$$

- Equations (a) and (b) hold for any $A \in \mathbb{R}^{d_y \times d_z}$, $b \in \mathbb{R}^{d_y}$, $C \in \mathbb{R}^{d_y \times d_y}$ symmetric positive definite, $m \in \mathbb{R}^{d_z}$, and $V \in \mathbb{R}^{d_z \times d_z}$ symmetric positive definite, where d_y and d_z are the dimensions of y and z, respectively.

- These equations can be interpreted as follows: if $Z \sim \mathcal{N}(m,V)$ and $Y \mid Z = z \sim \mathcal{N}(Az+b,C)$, then (a) tells us the marginal distribution of Y and (b) tells us the conditional distribution of $Z \mid Y = y$.

- (Bishop provides similar formulae in equations 2.113 - 2.117 on page 93, except that in (b) above, we have rewritten the right-hand side using the Woodbury matrix identity and equation C.5 of Bishop.)

4 Kalman filter (forward algorithm)

- Recall that the assumed model is:

$$\begin{aligned}
p(z_1) &= \mathcal{N}(z_1 \mid \mu_0, V_0)\\
p(z_j|z_{j-1}) &= \mathcal{N}(z_j \mid Fz_{j-1}, Q)\\
p(x_j|z_j) &= \mathcal{N}(x_j \mid Hz_j, R).
\end{aligned}$$

- In the Kalman filter, we compute the parameters of $p(z_j|x_{1:j})$ sequentially for $j = 1, \ldots, n$, in that order.

- First, consider $j = 1$.

$$\begin{aligned} p(z_1|x_1) &\underset{z_1}{\propto} p(x_1|z_1)p(z_1) = \mathcal{N}(x_1|Hz_1, R)\mathcal{N}(z_1|\mu_0, V_0) \\ &\underset{z_1}{\propto} \mathcal{N}\big(z_1 \mid \mu_0 + K_1(x_1 - H\mu_0),\ (I - K_1H)V_0\big) \end{aligned}$$

  by (b) with $y = x_1$, $z = z_1$, $A = H$, $b = 0$, $C = R$, $m = \mu_0$, and $V = V_0$, if we define

$$K_1 = V_0H^{\mathsf{T}}(HV_0H^{\mathsf{T}} + R)^{-1}.$$

- Therefore, $p(z_1|x_1) = \mathcal{N}(z_1|\mu_1, V_1)$ where

$$\begin{aligned} \mu_1 &= \mu_0 + K_1(x_1 - H\mu_0) \\ V_1 &= (I - K_1H)V_0. \end{aligned}$$

- Now, at the general step j, suppose that $p(z_{j-1}|x_{1:j-1}) = \mathcal{N}(z_{j-1}|\mu_{j-1}, V_{j-1})$ for some μ_{j-1}, V_{j-1}. Then

$$\begin{aligned} p(z_j|x_{1:j}) &\underset{z_j}{\propto} p(x_{1:j}, z_j) = \int p(x_{1:j}, z_{j-1}, z_j)dz_{j-1} \\ &= \int p(x_{1:j-1}, z_{j-1})p(z_j|z_{j-1})p(x_j|z_j)dz_{j-1} \\ &\underset{z_j}{\propto} \int \mathcal{N}(z_{j-1}|\mu_{j-1}, V_{j-1})\mathcal{N}(z_j|Fz_{j-1}, Q)\mathcal{N}(x_j|Hz_j, R)dz_{j-1} \\ &= \mathcal{N}(x_j|Hz_j, R)\int \mathcal{N}(z_j|Fz_{j-1}, Q)\mathcal{N}(z_{j-1}|\mu_{j-1}, V_{j-1})dz_{j-1} \\ &= \mathcal{N}(x_j|Hz_j, R)\mathcal{N}(z_j \mid F\mu_{j-1}, FV_{j-1}F^{\mathsf{T}} + Q) \end{aligned}$$

  by (a) with $y = z_j$, $z = z_{j-1}$, $A = F$, $b = 0$, $C = Q$, $m = \mu_{j-1}$, and $V = V_{j-1}$,

$$\underset{z_j}{\propto} \mathcal{N}(z_j \mid F\mu_{j-1} + K_j(x_j - HF\mu_{j-1}),\ (I - K_jH)V)$$

  by (b) with $y = x_j$, $z = z_j$, $A = H$, $b = 0$, $C = R$, $m = F\mu_{j-1}$, and $V = FV_{j-1}F^{\mathsf{T}} + Q$, if we define

$$K_j = VH^{\mathsf{T}}(HVH^{\mathsf{T}} + R)^{-1}.$$

- To avoid confusion, and for later reference, let's denote V by P_{j-1}, i.e.,

$$P_{j-1} = FV_{j-1}F^{\mathsf{T}} + Q.$$

- Therefore, $p(z_j|x_{1:j}) = \mathcal{N}(z_j|\mu_j, V_j)$, where

$$\begin{aligned} \mu_j &= F\mu_{j-1} + K_j(x_j - HF\mu_{j-1}) \\ V_j &= (I - K_jH)P_{j-1}. \end{aligned}$$

- Putting all this together, we have the following algorithm.

Kalman filter algorithm

Inputs: Observed data $x_1, \ldots, x_n$ and model parameters μ_0, V_0, F, Q, H, R.

1. Initialize:

$$\begin{aligned}
K_1 &= V_0 H^{\mathtt{T}} (H V_0 H^{\mathtt{T}} + R)^{-1} \\
\mu_1 &= \mu_0 + K_1(x_1 - H\mu_0) \\
V_1 &= (I - K_1 H) V_0 \\
P_1 &= F V_1 F^{\mathtt{T}} + Q.
\end{aligned}$$

2. For $j = 2, \ldots, n$:

$$\begin{aligned}
K_j &= P_{j-1} H^{\mathtt{T}} (H P_{j-1} H^{\mathtt{T}} + R)^{-1} \\
\mu_j &= F\mu_{j-1} + K_j(x_j - HF\mu_{j-1}) \\
V_j &= (I - K_j H) P_{j-1} \\
P_j &= F V_j F^{\mathtt{T}} + Q.
\end{aligned}$$

Then $p(z_j|x_{1:j}) = \mathcal{N}(z_j|\mu_j, V_j)$ for all $j = 1, \ldots, n$. Note that the algorithm can be used to perform real-time estimation of the current state as the data arrives, in an "online" fashion. Specifically, the mean μ_j can be used as an estimate of z_j, and our uncertainty in this estimate is quantified by V_j.

5 Rauch–Tung–Striebel smoother (backward algorithm)

- In the Rauch–Tung–Striebel smoother, we compute the parameters of $p(z_j|x_{1:n})$ sequentially for $j = n, n-1, \ldots, 1$, in that order.
- First, consider $j = n$. Actually, we already have the answer for this step, since we know that $p(z_n|x_{1:n}) = \mathcal{N}(z_n|\hat{\mu}_n, \hat{V}_n)$, where $\hat{\mu}_n = \mu_n$ and $\hat{V}_n = V_n$, with μ_n and V_n being the mean and covariance computed in the last step of the Kalman filter.
- At the general step j, suppose $p(z_{j+1}|x_{1:n}) = \mathcal{N}(z_{j+1}|\hat{\mu}_{j+1}, \hat{V}_{j+1})$. Then

$$p(z_j|x_{1:n}) = \int p(z_j|z_{j+1}, x_{1:n}) p(z_{j+1}|x_{1:n}) dz_{j+1}.$$

  We know $p(z_{j+1}|x_{1:n}) = \mathcal{N}(z_{j+1}|\hat{\mu}_{j+1}, \hat{V}_{j+1})$, but we need to do some work to get the parameters of $p(z_j|z_{j+1}, x_{1:n})$:

$$\begin{aligned}
p(z_j|z_{j+1}, x_{1:n}) &\underset{z_j}{\propto} p(z_j, z_{j+1}, x_{1:n}) = p(z_j, x_{1:j}) p(z_{j+1}|z_j) p(x_{j+1:n}|z_{j+1}) \\
&\qquad \text{(by the conditional independence properties of the model)} \\
&\underset{z_j}{\propto} p(z_{j+1}|z_j) p(z_j|x_{1:j}) = \mathcal{N}(z_{j+1}|F z_j, Q) \mathcal{N}(z_j|\mu_j, V_j) \\
&\qquad \text{(by the definition of } \mu_j \text{ and } V_j \text{ in the Kalman filter)}
\end{aligned}$$

$$\underset{z_j}{\propto} \mathcal{N}\big(z_j \,|\, \mu_j + K(z_{j+1} - F\mu_j),\, (I - KF)V_j\big)$$

by (b) with $y = z_{j+1}$, $z = z_j$, $A = F$, $b = 0$, $C = Q$, $m = \mu_j$, and $V = V_j$, if we define

$$K = V_j F^{\mathtt{T}}(FV_jF^{\mathtt{T}} + Q)^{-1}.$$

Note that $K = V_j F^{\mathtt{T}} P_j^{-1}$, by the definition of P_j in the Kalman filter.

- Now that we have a nice expression for $p(z_j|z_{j+1}, x_{1:n})$, we can plug it into the integral above to get

$$\begin{aligned} p(z_j|x_{1:n}) &= \int \mathcal{N}\big(z_j \,|\, \mu_j + K(z_{j+1} - F\mu_j),\, (I - KF)V_j\big)\mathcal{N}(z_{j+1}|\hat{\mu}_{j+1}, \hat{V}_{j+1})dz_{j+1} \\ &= \mathcal{N}\big(z_j \,|\, K\hat{\mu}_{j+1} + \mu_j - KF\mu_j,\, K\hat{V}_{j+1}K^{\mathtt{T}} + (I - KF)V_j\big) \end{aligned}$$

by (a) with $y = z_j$, $z = z_{j+1}$, $A = K$, $b = \mu_j - KF\mu_j$, $C = (I - KF)V_j$, $m = \hat{\mu}_{j+1}$, and $V = \hat{V}_{j+1}$,

$$= \mathcal{N}\big(z_j \,|\, \mu_j + K(\hat{\mu}_{j+1} - F\mu_j),\, V_j + K(\hat{V}_{j+1} - P_j)K^{\mathtt{T}}\big)$$

since $(I - KF)V_j = V_j - KP_jP_j^{-1}FV_j = V_j - KP_jK^{\mathtt{T}}$ (due to our earlier observation that $K = V_jF^{\mathtt{T}}P_j^{-1}$).

- To avoid confusion, let's denote K by C_j, i.e.,

$$C_j = V_jF^{\mathtt{T}}P_j^{-1}.$$

- Therefore, we have $p(z_j|x_{1:n}) = \mathcal{N}(z_j|\hat{\mu}_j, \hat{V}_j)$ where

$$\begin{aligned} \hat{\mu}_j &= \mu_j + C_j(\hat{\mu}_{j+1} - F\mu_j) \\ \hat{V}_j &= V_j + C_j(\hat{V}_{j+1} - P_j)C_j^{\mathtt{T}}. \end{aligned}$$

- Putting all this together, we get the following algorithm.

RTS smoother algorithm

Inputs: μ_j, V_j, and P_j for each $j = 1, \ldots, n$, computed by the Kalman filter algorithm.

1. Initialize: $\hat{\mu}_n = \mu_n$ and $\hat{V}_n = V_n$.
2. For $j = n-1, n-2, \ldots, 1$:

$$\begin{aligned} C_j &= V_jF^{\mathtt{T}}P_j^{-1} \\ \hat{\mu}_j &= \mu_j + C_j(\hat{\mu}_{j+1} - F\mu_j) \\ \hat{V}_j &= V_j + C_j(\hat{V}_{j+1} - P_j)C_j^{\mathtt{T}}. \end{aligned}$$

Then $p(z_j|x_{1:n}) = \mathcal{N}(z_j|\hat{\mu}_j, \hat{V}_j)$ for all $j = 1 \ldots, n$. The smoother algorithm can be interpreted as retrospectively improving the Kalman filter's estimate of the state z_j using the additional data observed at time steps $j + 1, \ldots, n$.