§ Filtering and smoothing

In quite many applications, we are concerned with estimating the latent variables from their observations / measurements. For example, estimating the trajectory of moving object based on its sensors readings (e.g., gyro and acceleration, angular velocity); estimating COVID cases from city waste water.

In our context, we model the latent variable by an SDE. More specifically, we consider a model

$$dX(t) = a(X(t))\,dt + b(X(t))\,dW(t), \quad X(0) = X_0,$$

$$Y_k \mid X(t_k) \sim P_{Y_k \mid X(t_k)}(y_k \mid x_k), \quad \text{We will abbreviate this as } p(y_k \mid x_k)$$

where the process X ~~at~~ stands for the latent / unobserved process, and the random variable Y_k represents the ~~actual~~ measurements at time t_k, which follows a conditional probability density function $p(y_k \mid x_k)$ given.

X2

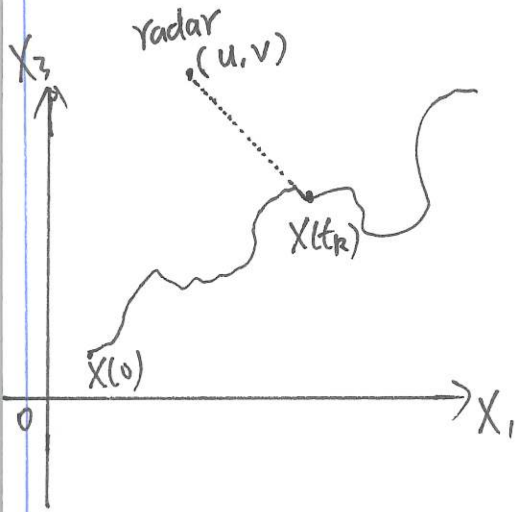
Example 28. Motion tracking model - 2D.

$$d\begin{bmatrix} X_1(t) \\ X_2(t) \\ X_3(t) \\ X_4(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} X_1(t) \\ X_2(t) \\ X_3(t) \\ X_4(t) \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} dW(t)$$

$W \in \mathbb{R}^2$

$$Y_k = \begin{bmatrix} \sqrt{(X_1(t_k) - u)^2 + (X_3(t_k) - v)^2} \\ \arctan\left(\frac{X_3(t_k) - v}{X_1(t_k) - u} \right) \end{bmatrix} + \xi_k,$$

$\xi_k \in \mathbb{R}^2$, $\xi_k \sim N(0, I_2)$



In this example, X_1 and X_3 represent the abscissa and ordinate of the object, respectively. The corresponding velocities are X_2 and X_4. We assume that there is a sensor/radar placed at (u, v) that measures the relative distance and bearing of the object. We would like to track estimate the object's state based on the measurements.

↳ Tracking in what sense?

X3

There are ~~of~~ various notions in estimating the latent process from the measurements. Recall the tracking example in Example 28, ~~we would like to~~ and suppose that at time t_{k-1} we have an estimate of the object. We would like to estimate the state of the object at time t_k based on the previous estimate at t_{k-1} and the measurement at the present time t_k. This gives the heuristics of the filtering problem, and this is formulated as follows.

§ Filtering

Suppose that we have a sequence of measurements $Y_{1:T} := \{Y_1, Y_2, \ldots Y_T\}$ at times $t_1 < t_2 < \cdots t_T$. The goal of the filtering is to estimate $X(t_k)$ progressively and sequentially for $k = 1, 2, \ldots$ from the measurements. More precisely, we would like to obtain the filtering probability density function

$$p(x_k \mid y_{1:k}), \quad \text{shorthand of } p_{X(t_k) \mid Y_{1:k}}(x_k \mid y_{1:k})$$

which represents the ~~condition~~ distribution of $X(t_k)$ conditionaly on the measurements up to time t_k. ~~Moreover, su~~ Furthermore, suppose that we know the previous filtering density $p(X_{k-1} | y_{1:k-1})$ we want to express $p(X_k | y_{1:k})$ in terms of $p(X_{k-1} | y_{1:k-1})$, so that the filtering algorithm is an online recursive routine

$$p(X_0) \to p(X_1 | y_{1:1}) \to p(X_2 | y_{1:2}) \to \cdots p(X_k | y_{1:k}) \cdots$$

We now show how to do so by the Markov property and Baye's rule.

By Baye's rule

$$p(X_k | y_{1:k}) = p(X_k | y_{1:k-1}, y_k) = \frac{p(y_k | X_k)\, p(X_k | y_{1:k-1})}{p(y_k | y_{1:k-1})}$$

$$= \frac{p(y_k | X_k)\, p(X_k | y_{1:k-1})}{\int p(y_k | X_k)\, p(X_k | y_{1:k-1})\, dX_k},$$

where

predictive density

$$\underbrace{p(X_k | y_{1:k-1})} = \int \underbrace{p(X_k | X_{k-1})}_{\text{transition density}} \overbrace{p(X_{k-1} | y_{1:k-1})}^{\text{the previous filtering density}} dX_{k-1}$$

It is then clear that we can use $p(X_{k-1} | y_{1:k-1})$ to compute $p(X_k | y_{1:k-1})$ then ~~use the prediction~~ correct the prediction $p(X_k | y_{1:k-1})$ based on $p(y_k | X_k)$ to yield $p(X_k | y_{1:k})$.

X5

Algorithm 29 Filter

Input : $y_{1:T}$, and initial condition $p(x_0)$

Define $p(x_0 | y_{1:0}) := p(x_0)$

output : $p(x_1 | y_{1:1})$, $p(x_2 | y_{1:2})$, .. $p(x_T | y_{1:T})$

1. For $k = 1, 2, \dots T$ do

2. $$p(x_k | y_{1:k-1}) = \int p(x_k | x_{k-1}) \, p(x_{k-1} | y_{1:k-1}) \, dx_{k-1}$$

3. $$p(x_k | y_{1:k}) = \frac{p(y_k | x_k) \, p(x_k | y_{1:k-1})}{\int p(y_k | x_k) \, p(x_k | y_{1:k-1}) \, dx_k}$$

Return $p(x_1 | y_{1:1})$, $p(x_2 | y_{1:2})$,.. $p(x_T | y_{1:T})$

$O(T)$ complexity

There is also another notion of the estimation, called smoothing, which is widely used as well especially in machine learning. The smoothing refers to estimate the state based on all the measurements unlike the filtering which uses only the historical measurements. More specifically, the smoothing density is

$$p(x_k | y_{1:T}) \quad , \quad \text{for } k = 1, 2, \dots T.$$

and we can compute it by

Previous smoothing density at t_{k+1}

$$p(x_k | y_{1:T}) = \int p(x_k | x_{k+1}, y_{1:T}) \, \underline{p(x_{k+1} | y_{1:T})} \, dx_{k+1}$$

Thanks to the Markov property, we have

$$p(x_k \mid x_{k+1}, y_{1:T}) = p(x_k \mid x_{k+1}, y_{1:k})$$

$$= \frac{p(x_{k+1} \mid x_k, y_{1:k})\, p(x_k \mid y_{1:k})}{p(x_{k+1} \mid y_{1:k})}$$

$$= \frac{p(x_{k+1} \mid x_k)\, p(x_k \mid y_{1:k})}{p(x_{k+1} \mid y_{1:k})}$$

Substitute the expression for $p(x_k \mid x_{k+1}, y_{1:T})$ back to $p(x_k \mid y_{1:T})$ we arrive at

$$p(x_k \mid y_{1:T}) = p(x_k \mid y_{1:k}) \int \frac{p(x_{k+1} \mid x_k)\, p(x_{k+1} \mid y_{1:T})}{p(x_{k+1} \mid y_{1:k})} dx_{k+1} .$$

Hence, to obtain the smoothing density at any $t_1 \le t_k \le t_T$, we can first run a filter forward to get $p(x_1 \mid y_1)$, $p(x_2 \mid y_{1:2})$, ... $p(x_T \mid y_{1:T})$, then backward compute $p(x_{T-1} \mid y_{1:T})$, $p(x_{T-2} \mid y_{1:T})$ ... $p(x_{k+1} \mid y_{1:T})$, $p(x_k \mid y_{1:T})$. This is summarised in the following algorithm.

ZZ

Algorithm 29. Smoothing

Input: $y_{1:T}$, $p(x_0)$

output: $p(x_1 | y_{1:T})$, $p(x_2 | y_{1:T})$, ... $p(x_T | y_{1:T})$

1. Run the filter to obtain
   $p(x_1 | y_1)$, $p(x_2 | y_{1:2})$, ... $p(x_T | y_{1:T})$
2. For $k = T-1, T-2, \dots 1$ do
   $$p(x_k | y_{1:T}) = p(x_k | y_{1:k}) \int \frac{p(x_{k+1} | x_k)\, p(x_{k+1} | y_{1:T})}{p(x_{k+1} | y_{1:k})} dx_{k+1}$$

Return $p(x_1 | y_{1:T})$, $p(x_2 | y_{1:T})$, ... $p(x_T | y_{1:T})$

Remark: the predictiven density is already computed in the filtering routine. No need to comput it again.

Unfortunately, the filtering and smoothing algorithms are in reality not ~~impen~~ implementable, because 1) the transition density $p(x_k | x_{k-1})$ is intractable. 2) the computations for the integrals are intractable (e.g., $\int p(y_k | x_k)\, p(x_k | y_{1:k-1})\, dx_k$). Therefore, in practice we have to approximate the filtering and smoothing densities, except for a few isolated cases, suchas Linear Gaussian systems. ~~In wha~~

In what follows, we introduce the celebrated Kalman filter and Rauch-Tung-Striebel smoother.
RTS

§ Kalman filter and RTS smoother

The Kalman filters and RTS smoothers ~~works~~ work on Linear SDEs and linear Gaussian measurement models. More specifically, the models are of the form

$$dX(t) = A\, X(t)\, dt + B\, dW(t),$$

$$Y_k = H\, X(t_k) + \xi_k, \quad \xi_k \sim N(0, \Xi_k),$$

where $X \in \mathbb{R}^d$, $A \in \mathbb{R}^{d \times d}$, $B \in \mathbb{R}^{d \times w}$, $H \in \mathbb{R}^{s \times d}$, $\xi_k \in \mathbb{R}^s$. Of course we can also let the SDE coefficients A and B be time-dependent (see, Lecture 5), but for simplicity, let us consider them time-invariant.

As we have seen in Lecture 5 that solution to the linear SDE is Gaussian process, hence, the measurement random variables Y_k's are Gaussian too. Consequently, all the

PDFs in the filtering and smoothing algorithms are Gaussian. Hence, for this model, we can solve the filtering and smoothing problems in closed-form by playing with "Gaussian algebras". The most important algebra we are going to use is the Gaussian identity.

Remark 30. Gaussian identity. Let $\alpha \in \mathbb{R}^{d_\alpha}$ and $\beta \in \mathbb{R}^{d_\beta}$ be two jointly Normal distributed vectors, such that

$$\begin{bmatrix} \alpha \\ \beta \end{bmatrix} \sim N\left(\begin{bmatrix} \mathbb{E}[\alpha] \\ \mathbb{E}[\beta] \end{bmatrix}, \begin{bmatrix} \mathrm{Cov}[\alpha], & \mathrm{Cov}[\alpha, \beta] \\ \mathrm{Cov}[\beta, \alpha] & \mathrm{Cov}[\beta] \end{bmatrix} \right).$$

Then the distribution of α condition on β is also Normal with mean

$$\mathbb{E}[\alpha \mid \beta] = \mathbb{E}[\alpha] + \mathrm{Cov}[\alpha, \beta] \left(\mathrm{Cov}[\beta] \right)^{-1} \left(\beta - \mathbb{E}[\beta] \right)$$

and covariance

$$\mathrm{Cov}[\alpha \mid \beta] = \mathrm{Cov}[\alpha] - \mathrm{Cov}[\alpha, \beta] \left(\mathrm{Cov}[\beta] \right)^{-1} \mathrm{Cov}[\beta, \alpha].$$

~~Mutadis mutantis~~ for β condition on α.
Mutatis mutandis

We are now ready to derive the filtering densities. Since ~~the~~ they are Gaussian, denote $p(x_k \mid y_{1:k}) := N(x_k \mid m_k, V_k)$ ~~for~~, where m_k and V_k stand for the filtering mean and covariance, respectively. ~~Also~~ Also remark that m_k ~~and V_k~~ depend on the data $y_{1:k}$. V_k does not depend on $y_{1:k}$ ~~Since~~ The first step in the filtering Algorithm 29 requires to compute

$$p(x_k \mid y_{1:k-1}) = \int p(x_k \mid x_{k-1})\, p(x_{k-1} \mid y_{1:k-1})\, dx_{k-1}.$$

The properties of the linear SDE tell us that

$$\begin{aligned} p(x_k \mid x_{k-1}) &= N(x_k \mid \mathbb{E}[X(t_k) \mid X(t_{k-1})], \mathrm{Cov}[X(t_k) \mid X(t_{k-1})]) \\ &= N(x_k \mid F_k x_{k-1}, \Sigma_k). \end{aligned}$$

please recall how to compute F_k and Σ_k in Lecture 5.

Hence,

$$\begin{aligned} p(x_k \mid y_{1:k-1}) &= \int N(x_k \mid F_k x_{k-1}, \Sigma_k)\, N(x_{k-1} \mid m_{k-1}, V_{k-1})\, dx_{k-1} \\ &= N(x_k \mid F_k m_{k-1}, F_k V_{k-1} F_k^T + \Sigma_k) \\ &:= N(x_k \mid m_k^-, V_k^-) \end{aligned}$$

shorthand notations

Next, we apply the Gaussian identity in Remark 30 to obtain $p(x_k \mid y_{1:k})$. Now imagine that the α an β in Remark 30 are $X_k \mid y_{1:k-1}$ and $Y_k \mid y_{1:k-1}$, respectively, then

$$m_k := \mathbb{E}[X_k \mid y_{1:k}] = \mathbb{E}[X_k \mid y_{1:k-1}] + \mathrm{Cov}[X_k, Y_k \mid y_{1:k-1}] \, \mathrm{Cov}[Y_k \mid y_{1:k-1}]^{-1} \, (y_k - \mathbb{E}[Y_k \mid y_{1:k-1}]),$$

$$V_k := \mathrm{Cov}[X_k \mid y_{1:k}] = \mathrm{Cov}[X_k \mid y_{1:k-1}] - \mathrm{Cov}[X_k, Y_k \mid y_{1:k-1}] \, \mathrm{Cov}[Y_k \mid y_{1:k-1}]^{-1} \, \mathrm{Cov}[Y_k, X_k \mid y_{1:k-1}].$$

Recall that

$$\begin{aligned} p(y_k \mid y_{1:k-1}) &= \int p(y_k \mid x_k)\, p(x_k \mid y_{1:k-1})\, dx_k \\ &= \int N(y_k \mid H x_k, \Xi_k)\, N(x_k \mid m_k^-, V_k^-)\, dx_k \\ &= N(y_k \mid H m_k^-, H V_k^- H_k^T + \Xi_k) \end{aligned}$$

and that

$$\begin{aligned} \mathrm{Cov}[X_k, Y_k \mid y_{1:k-1}] &= \mathrm{Cov}[X_k, H X_k + \xi_k \mid y_{1:k-1}] \\ &= V_k^- H^T. \end{aligned}$$

Putting everything together we have

$$m_k = m_k^- + V_k^- H^T (H V_k^- H^T + \Sigma_k)^{-1} (y_k - H m_k^-)$$

$$V_k = V_k^- - V_k^- H^T (H V_k^- H^T + \Sigma_k)^{-1} H V_k^- .$$

We ~~can~~ summarise the kalman fitering in the following algorithm.

Algorithm 3.1 Kalman filter.

Input: Measurements $y_{1:T}$, initial mean m_0 and covariance V_0

output: $\{m_k, V_k\}_{k=1}^T$

1. For $k = 1, 2, \ldots T$ do
2. Compute F_k and Σ_k // can be pre-computed
3. $m_k^- = F_k m_{k-1}$
4. $V_k^- = F_k V_{k-1} F_k^T + \Sigma_k$ — prediction
5. $S_k = H_k V_k^- H_k^T + \Sigma_k$
6. Kalman gain $K_k = V_k^- H_k^T S_k^{-1}$
7. $m_k = m_k^- + K_k (y_k - H_k m_k^-)$
8. $V_k = V_k^- - K_k S_k K_k^T$ — update

Return $\{m_k, V_k\}_{k=1}^T$

Note that V_k does Not depend on the data values $y_{1:k}$.

By using a similar derivation, we can obtain the RTS smoother for the smoothing densities. This is left as a homework for the readers. The RTS smoother is summarised as follows.

Denote $p(x_k \mid y_{1:T}) := N(x_k \mid m_k^s, V_k^s)$

Algorithm 82. RTS smoother

Input: Measurements $y_{1:T}$, initial mean m_0 and covariance V_0.

Output: $\{m_k^s, V_k^s\}_{k=1}^T$

1. Do the Kalman filter to obtain $\{m_k, V_k\}_{k=1}^T$
2. Let $m_T^s = m_T$ and $V_T^s = V_T$
3. For $k = T-1, T-2, \dots 1$ do

$$m_{k+1}^- = F_{k+1} m_k$$

$$V_{k+1}^- = F_{k+1} V_k F_{k+1}^T + \Sigma_{k+1}$$

these two predictive quantities are already computed in the filter. no need to comput them again.

$$G_k = V_k F_{k+1}^T (V_{k+1}^-)^{-1}$$ smoother gain

$$m_k^s = m_k + G_k(m_{k+1}^s - m_{k+1}^-)$$

$$V_k^s = V_k + G_k(V_{k+1}^s - V_{k+1}^-) G_k^T$$

Return $\{m_k^s, V_k^s\}_{k=1}^T$

Example. 33. Consider a simple motion model

$$d\begin{bmatrix} X_1(t) \\ X_2(t) \\ X_3(t) \\ X_4(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} X_1(t) \\ X_2(t) \\ X_3(t) \\ X_4(t) \end{bmatrix} dt + \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} dw(t)$$

$$Y_k = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} X(t_k) + \xi_k, \quad \xi_k \sim N(0, \Xi)$$

$X(0) \sim N(0, I_4)$.

1) Generate a pair of trajectory and measurements $(X_{1:T}, Y_{1:T})$ at times $t_1, t_2, \ldots t_T$

2) Use Kalman filter and RTS smoother to estimate $X_{1:T}$ from $Y_{1:T}$. That is, comput the filtering and smoothing densities at the times.

See, `lec6_kalman_filter_rts_smoother.ipynb` for how to do so.