§ 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

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

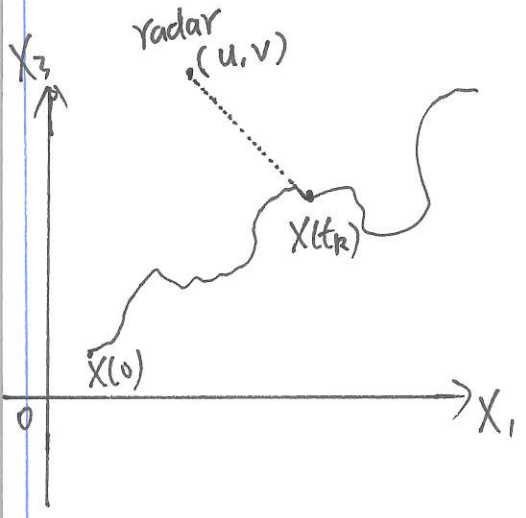
$$Y_k \mid X(t_k) \sim P_{Y_k|X(t_k)}(y_k \mid X_k).$$ We will abbreviate this as $P(y_k \mid X_k)$

where the process X ~~that~~ 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.

Example 28. Motion tracking model - 2D.

$$d\begin{bmatrix} X_1(t) \\ X_2(t) \\ X_3(t) \\ X_4(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} X_1(t) \\ X_2(t) \\ X_3(t) \\ X_4(t) \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} dW(t) \qquad W \in \mathbb{R}^2$$

$$Y_k = \begin{bmatrix} \sqrt{(X_1(t_k)-u)^2 + (X_3(t_k)-v)^2} \\ \arctan\left(\frac{X_3(t_k)-v}{X_1(t_k)-u}\right) \end{bmatrix} + \xi_k, \qquad \xi_k \in \mathbb{R}^2,\ \xi_k \sim N(0, I_2)$$



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 estimate (track) the object's state based on the measurements.

↳ Tracking in what sense?

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

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

which represents the ~~condition~~ distribution of $X(t_k)$ conditionally 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 $p(x_k | y_{1:k-1})$ (predictive density) $= \int \underbrace{p(x_k | x_{k-1})}_{\text{transition density}} \underbrace{p(x_{k-1} | y_{1:k-1})}_{\text{the previous filtering density}} dx_{k-1}$

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 $p(x_k | y_{1:k})$.

Algorithm 29 Filter

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

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

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

1. For $k = 1, 2, \ldots T$ 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 $p(x_1 | y_{1:1}), p(x_2 | y_{1:2}), \ldots p(x_T | y_{1:T})$

$O(T)$ 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

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

and we can compute it by

Previous smoothing density at t_{k+1}

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

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 summarized in the following algorithm.

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 $k = T-1, T-2, \ldots 1$ 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 predictive density is already computed in the filtering routine. No need to compute 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, such as Linear Gaussian systems. ~~In this~~

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

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

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

$$\mathbb{E}[\alpha | \beta] = \mathbb{E}[\alpha] + Cov[\alpha, \beta](Cov[\beta])^{-1}(\beta - \mathbb{E}[\beta])$$

and covariance

$$Cov[\alpha | \beta] = Cov[\alpha] - Cov[\alpha, \beta](Cov[\beta])^{-1} Cov[\beta, \alpha].$$

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

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

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

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

Hence,

$$p(x_k \mid y_{1:k-1}) = \int N(x_k \mid F_k x_{k-1}, \Sigma_k)\, N(x_{k-1} \mid m_{k-1}, V_{k-1})\, dx_{k-1}$$
$$= N(x_k \mid F_k m_{k-1}, F_k V_{k-1} F_k^T + \Sigma_k)$$
$$:= N(x_k \mid \bar{m}_k, \bar{V}_k)$$ 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 and 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

$$p(y_k \mid y_{1:k-1}) = \int p(y_k \mid x_k)\, p(x_k \mid y_{1:k-1})\, dx_k$$

$$= \int N(y_k \mid H x_k, \Sigma_k)\, N(x_k \mid m_k^-, P_k^-)\, dx_k$$

$$= N(y_k \mid H m_k^-, H V_k^- H_k^T + \Sigma_k)$$

and that

$$\mathrm{Cov}[x_k, y_k \mid y_{1:k-1}] = \mathrm{Cov}[x_k, H x_k + \epsilon_k \mid y_{1:k-1}]$$

$$= V_k^- H^T.$$

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 summarise the kalman fitering in the following algorithm.

Algorithm 3 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. $m_k^- = F_k m_{k-1}$ } prediction
4. $V_k^- = F_k V_{k-1} F_k^T + \Sigma_k$ }
5. $S_k = H_k V_k^- H_k^T + \Sigma_k$
6. Kalman gain $K_k = V_k^- H_k^T S_k^{-1}$ } update
7. $m_k = m_k^- + K_k (y_k - H_k m_k^-)$ }
8. $V_k = V_k^- - K_k S_k K_k^T$ }

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 $k = T-1, T-2, \ldots 1$ do

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

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

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

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

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

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

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

Example. 33. Consider a simple motion model

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

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

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

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, compute the filtering and smoothing densities at the times.

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

Parameter estimation with maximum likelihood

In Lecture 4, we have already talked about the parameter estimation of SDEs. However, we assummed that we can directly observe the SDE. But, in reality, the SDE is usually a latent process, which we do not directly observe. Consider a model as follows.

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

$$Y_k \mid X(t_k) \sim p(y_k \mid x_k;\theta).$$

where $\theta \in \mathbb{R}^{d_\theta}$ is an unknow parameter that we want to estimate. from the measurements ~~$\{y_1, y_2, \dots, y_T\} =: y_{1:T}$~~ $y_{1:T} := \{y_1, y_2, \dots, y_T\}$. To estimate θ, we can use the maximum likelihood method, and the likelihood is

$$p(y_{1:T};\theta) = \int \prod_{k=1}^{T} p(y_k \mid x_k;\theta) \prod_{k=1}^{T} p(x_k \mid x_{k-1};\theta)\, p(x_0)\, dx_{1:T}. \quad (1)$$

Remark that compared to the parameter estimation in Lecture 4, we now need to integrate out the variables $x_{1:T}$ in order to

Compute $p(y_{1:T};\theta)$ which is way harder ~~than that~~, and is not analytically tractable in general because of the intractable transition density $p(x_k \mid x_{k-1})$.

The equation in ① is not the only way to express the likelihood. We can also factorise $p(y_{1:T};\theta)$ in terms of the filtering densities:

$$p(y_{1:T};\theta) = \prod_{k=1}^{T} p(y_k \mid y_{1:k-1};\theta)$$

$$= \prod_{k=1}^{T} \int p(y_k \mid x_k;\theta)\, p(x_k \mid y_{1:k-1};\theta)\, dx_k$$

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

Hence we can compute $p(y_{1:T};\theta)$ along with the run of the filter.

Example 34.

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

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

87

$$p(y_{1:T};\theta) = \prod_{k=1}^{T} \int p(y_k \mid x_k;\theta)\, p(x_k \mid y_{1:k-1};\theta)\, dx_k$$

$$= \prod_{k=1}^{T} \int N(y_k \mid H(\theta)x_k, \Xi)\, N(x_k \mid m_k^-(\theta), V_k^-(\theta))\, dx_k$$

$$= \prod_{k=1}^{T} N(y_k \mid H(\theta)m_k^-(\theta),\ H(\theta)V_k^-H(\theta)^T + \Xi).$$

Hence the objective function for carrying out the MLE is

$$-\log p(y_{1:T};\theta) = \frac{1}{2}\sum_{k=1}^{T}\Big((y_k - H(\theta)m_k^-(\theta))^T S_k(\theta)^{-1}(y_k - H(\theta)m_k^-(\theta)) + \log\det(2\pi S_k(\theta))\Big).$$

Algorithm 35. Computing the negative log likelihood with Kalman filter

Input: The same as with Algorithm 31.

Output: The same as with Algorithm 31.

1. The same as with Algorithm 31, except that additionally we compute

$$\text{nell}_k = \frac{1}{2}\Big((y_k - H_k m_k^-)^T S_k^{-1}(y_k - H_k m_k^-) + \log\det(2\pi S_k)\Big)$$

at the end of each loop.

Return $\sum_{k=1}^{T} \text{nell}_k$.

§ Non-linear filtering and smoothing

In the previous section, we have seen how to use Kalman filters and smoothers which are optimal in the least square sense for linear Gaussian models.

However, in reality, we often have to tackle non-linear models. Consider models of the form:

$$dX(t) = a(X(t))\,dt + b(X(t))\,dW(t), \quad X(0) = X_0 \sim p(x_0)$$

$$Y_k \mid X(t_k) \sim p(y_k \mid x_k).$$

It is known challenging to implement Algorithms 28 and 29 to solve the filtering and smoothing densities $p(x_k \mid y_{1:k})$ and $p(x_k \mid y_{1:T})$ in closed-form. Hence, we may have to approximate them. The most celebrated approximations are arguably the

Note: if the measurement model is continuous in time, e.g. $dY(t) = \int_0^t h(X(s))\,ds + \int_0^t g(s)\,dW_2(s)$, the filtering density is governed by the Kushner-Stratonovich equation which is a stochastic partial differential equation. We can simulate such an SPDE to solve the filtering problem in continuous-time.

Gaussian filters and smoothers, and the particle filters and smoothers. ~~The Gaussian filters include, for instance, the extended K~~ For simplicity, in this lecture, we introduce the Gaussian filters and smoothers.

Remark: the backbone of particle filters and smoothers is the sequential Monte Carlo techniques. Essentially, insted of trying to approximate the filtering density $p(x_k | y_{1:k})$, the particle filters approximate any expectation $\mathbb{E}[\varphi(x_{0:k}) | y_{1:k}]$ for arbitary function φ by Monte Carlo. You can also think that $p(x_k | y_{1:k})$ is represented by a bunch of weighted samples. See, Chopin and Papaspiliopoulos, 2020.

↳ Gaussian approximate filtering and smoothing

please avoid calling them Gaussian assummed density filters and smoothers, since you cannot assume something that is not true.

The ~~idea~~ essence of the approximate Gaussian filters and smoothers consists in the following approximations:

1) $p(x_k | y_{1:k}) \approx \mathcal{N}(x_k | m_k, V_k)$ Gaussian filtering density

2) $p(x_k | y_{1:T}) \approx \mathcal{N}(x_k | m_k^s, V_k^s)$ Gaussian smoothing density

3) $p(x_k | x_{k-1}) \approx N(x_k | \mathbb{E}[x(t_k) | x(t_{k-1})], \mathrm{Cov}[x(t_k) | x(t_{k-1})])$ Gaussian transition

4) $p(x_0) \approx N(x_0 | m_0, V_0)$ Gaussian initial

5) $p(y_k | x_k) \approx N(y_k | h(x_k), \Sigma_k)$ $y_k = h(x(t_k)) + h_k$, $h_k \sim N(0, \Sigma_k)$

By apply these Gaussian approximations, we can solve the filtering and smoothing problems in closed-form by playing with Gaussian algebras. Substituting these approximations back into the general filtering and smoothing equations in Algorithms ~~3~~ 8 and 29, and use a similar derivation for the Kalman fiter and RTS smoother, we have the ~~so~~ skeleton of Gaussian approximate filter and smoother.

Algorithm 36 Gaussian approximate fiter.

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

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

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

① predictions

$$\bar{m_k} \approx \int x_k \, p(x_k | y_{1:k-1}) \, dx_k$$

$$\bar{V_k} \approx \int (x_k - \bar{m_k})(x_k - \bar{m_k})^T p(x_k | y_{1:k-1}) \, dx_k$$

update

② $\begin{cases} \mu_k = \int h(x_k^-)\, N(x_k^- \mid m_k^-, V_k^-)\, dx_k^- \\ S_k = \int (h(x_k^-) - \mu_k)(h(x_k^-) - \mu_k)^T N(x_k^- \mid m_k^-, V_k^-)\, dx_k^- + \Xi_k \\ K_k = \left(\int (x_k^- - m_k^-)(h(x_k^-) - \mu_k)^T dx_k^- \right) S_k^{-1} \\ m_k = m_k^- + K_k (y_k - \mu_k) \\ V_k = V_k^- + K_k S_k K_k^T \end{cases}$

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

Algorithm 3.7 Gaussian approximate smoothing.

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

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

1. Do the Gaussian filtering to obtain $\{m_k, V_k\}_{k=1}^T$ and $\{m_k^-, V_k^-\}_{k=2}^T$

2. Let $m_T^s = m_T$ and $V_T^s = V_T$

3. For $k = T-1, T-2, \ldots 1$ do

$D_{k+1} \approx \mathrm{Cov}[x_k, x_{k+1} \mid y_{1:k}]$.

$D_{k+1} = \int x_k \mathbb{E}[x(t_{k+1}) \mid x_k]\, N(x_k \mid m_k, V_k)\, dx_k - m_k (m_{k+1}^-)^T$

$G_k = D_{k+1} (V_{k+1}^-)^{-1}$

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

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

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

The Gaussian filter and smoother look very similar to the Kalman filter and smoother, except that now we have to solve some Gaussian expectations with non-linear integrands.

These expectations specifically refer to:

① $\begin{cases} m_k^- \triangleq \int x_k \, p(x_k \mid y_{1:k-1}) \, dx_k \\ V_k^- \triangleq \int (x_k - m_k^-)(x_k - m_k^-)^T p(x_k \mid y_{1:k-1}) \, dx_k \end{cases}$

② : m_k, S_k, K_k

③ : $D_{k+1} = \int x_k \, \mathbb{E}[x(t_{k+1}) \mid x_k] \, N(x_k \mid m_k, V_k) \, dx_k - m_k (m_{k+1}^-)^T$.

How do we compute the predictive mean and covariance in ①? There are a plenty of ways to compute them.

For example, recall Lecture 4, the statistics of SDEs, we have

$$\frac{d\,\mathbb{E}[x(t) \mid y_{1:k-1}]}{dt} = \mathbb{E}[a(x(t)) \mid y_{1:k-1}]$$

$$\frac{d\,\mathrm{Cov}[x(t) \mid y_{1:k-1}]}{dt} = \mathbb{E}[a(x(t))(x(t) - \hat{m}(t))^T + (x(t) - \hat{m}(t)) a(x(t))^T + b(x(t)) b(x(t))^T \mid y_{1:k-1}]$$

$t \in [t_{k-1}, t_k]$

denote $\hat{m}(t) := \mathbb{E}[x(t) \mid y_{1:k-1}]$, $\hat{P}(t) := \mathrm{Cov}[x(t) \mid y_{1:k-1}]$

shorthand

Hence, to obtain m_k^- and V_k^-, we solve the ODE above at t_k starting from the initial values m_{k-1} and V_{k-1} at t_{k-1}. But how

do we solve the expectation on the RHS of the ODEs?

We can approximate the expectation by Gaussian, then

$$\mathbb{E}[a(x(t)) \mid y_{1:k-1}] \approx \int a(x(t))\, \mathcal{N}(x(t) \mid \mathring{m}(t), \mathring{V}(t))\, dx(t)$$

we can further approximate a by Linearisation

$$a(x) \approx a(\mathring{m}) + \boxed{Ja(\mathring{m})}(x - \mathring{m}) + \cdots,$$

Jacobian of a at $\mathring{m}$

then we have

$$\mathbb{E}[a(x(t)) \mid y_{1:k-1}] \approx \mathbb{E}[a(\mathring{m}(t)) + Ja(\mathring{m}(t))(x(t) - \mathring{m}(t))]$$
$$= a(\mathring{m}(t))$$

$$\mathbb{E}[a(x(t))(x(t) - \mathring{m}(t))^T + (x(t) - \mathring{m}(t))a(x(t))^T + b(x(t))b(x(t))^T \mid y_{1:k-1}]$$
$$\approx \mathring{V}(t)\, Ja(\mathring{m}(t))^T + Ja(\mathring{m}(t))\mathring{V}(t) + b(\mathring{m}(t))b(\mathring{m}(t))^T.$$

putting everything together, we have

$$\begin{cases} \frac{d\mathring{m}(t)}{dt} = a(\mathring{m}(t)) \\ \frac{d\mathring{V}(t)}{dt} = \mathring{V}(t)\, Ja(\mathring{m}(t))^T + Ja(\mathring{m}(t))\mathring{V}(t) + b(\mathring{m}(t))b(\mathring{m}(t))^T \end{cases}, \quad t \in [t_{k-1}, t_k].$$

starting from $\mathring{m}(t_{k-1}) = m_{k-1}$ and $\mathring{V}(t_{k-1}) = V_{k-1}$, and we solve the ODE at t_k to get $\mathring{m}(t_k) = m_k^-$ and $\mathring{V}(t_k) = V_k^-$.

Solve the ODEs with e.g., Euler, Runge-Kutta.

This gives a continuous-discrete extended Kalman filter, which consists in solving a system of non-linear ODEs in the prediction step.

We can also solve ① in a different flavour. Recall:

$$\int \chi_k \, p(\chi_k \mid y_{1:k-1}) \, d\chi_k$$

$$= \int \chi_k \, p(\chi_k \mid \chi_{k-1}) \, p(\chi_{k-1} \mid y_{1:k-1}) \, d\chi_k \, d\chi_{k-1}$$

$$m_k^- \approx \int \mathbb{E}[X(t_k) \mid X(t_{k-1}) = \chi_{k-1}] \, \mathcal{N}(\chi_{k-1} \mid m_{k-1}, V_{k-1}) \, d\chi_{k-1}$$

a Non-Linear integrand.

$$\int (\chi_k - m_k^-)(\chi_k - m_k^-) \, p(\chi_k \mid y_{1:k-1}) \, d\chi_k$$

$$V_k^- \approx \int \mathbb{E}[(X(t_k) - m_k^-)(X(t_k) - m_k^-)^T \mid X(t_{k-1}) = \chi_{k-1}] \, \mathcal{N}(\chi_{k-1} \mid m_{k-1}, V_{k-1}) \, d\chi_{k-1}$$

~~$\approx \mathrm{Cov}[X(t_k) \mid X(t_{k-1}) = m_{k-1}]$.~~

Denote $f(x) := \mathbb{E}[X(t_k) \mid X(t_{k-1}) = x]$, then we can linearise f at m_{k-1} by $f(x) \approx f(m_{k-1}) + \boxed{Jf}(m_{k-1})(x - m_{k-1}) + \dots$. Then

Jacobian of f.

we have

$$m_k^- = f(m_{k-1}) = \mathbb{E}[X(t_k) \mid X(t_{k-1}) = m_{k-1}],$$

$$V_k^- = Jf(m_{k-1}) V_{k-1} Jf(m_{k-1})^T + \mathrm{Cov}[X(t_k) \mid X(t_{k-1}) = m_{k-1}].$$

How V_k^- is arrived? Law of total covariance

This gives another version of the extended Kalman filter.

Similarly, we can also apply the linearisation to solve for M_k, S_k, and K_k in ② ~~and~~. We linearise $h(x) \approx h(m_k^-) + Jh(m_k^-)(x - m_k^-)$. Then we have

Jacobian of h.

$$M_k = \int h(x_k^-) N(x_k^- | m_k^-, V_k^-) dx_k^-$$
$$\approx h(m_k^-)$$
$$S_k = Jh(m_k^-) V_k^- Jh(m_k^-)^T + \hat{\Sigma}_k$$
$$K_k = V_k^- Jh(m_k^-)^T S_k^{-1}.$$

We ~~can~~ sumarise the extended Kalman filter in the following algorithm.

this another EKF.

Algorithm 38 Extended Kalman filter.

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

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

~~1. Denote $f(x) := \mathbb{E}[x(t_k) | x(t_{k-1}) = x]$.~~

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

denote $f(x) := \mathbb{E}[x(t_k) | x(t_{k-1}) = x]$.

$$m_k^- = \mathbb{E}[x(t_k) | x(t_{k-1}) = m_{k-1}]$$
$$F_k = Jf(m_{k-1})$$

$$\Sigma_k = \mathrm{Cov}[X(t_k) \mid X(t_{k-1}) = m_{k-1}].$$

$$V_k^- = F_k V_{k-1} F_k^T + \Sigma_k.$$

Recall Lecture 3 and 4 for how to compute the conditional mean and covariance. E.g., Euler–Maruyama.

$$H_k = Jh(m_k^-)$$

$$S_k = H_k V_k^- H_k^T + \Xi_k$$

$$K_k = V_k^- H_k^T S_k^{-1}$$

$$m_k = m_k^- + K_k(y_k - h(m_k^-))$$

$$V_k = V_k^- - K_k S_k K_k^T$$

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

Mutatis Mutandis, we can derive the extended Kalman smoother

Algorithm 39. Extended Kalman smoother

Input: $y_{1:T}$, m_0, V_0

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

1. Do the extended Kalman filter to obtain $\{m_k, V_k\}_{k=1}^T$ and $\{m_k^-, V_k^-\}_{k=2}^T$.

2. Let $m_T^S = m_T$ and $V_T^S = V_T$

3. For $k = T-1, T-2, \ldots 1$ do

$$F_{k+1} = Jf(m_k)$$

$$G_k = V_k F_{k+1}^T (\bar{V}_{k+1})^{-1}$$

$$m_k^s = m_k + G_k (m_{k+1}^s - \bar{m}_{k+1})$$

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

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

Example 40.

$$d\begin{bmatrix} X_1(t) \\ X_2(t) \\ X_3(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ -\left(\frac{\sqrt{3}}{g(X_3(t))}\right)^2 & \frac{-2\sqrt{3}}{g(X_3(t))} & 0 \\ 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} X_1(t) \\ X_2(t) \\ X_3(t) \end{bmatrix} dt + \begin{bmatrix} 0 & 0 \\ 2\left(\frac{\sqrt{3}}{g(X_3(t))}\right)^{\frac{3}{2}} & 0 \\ 0 & 1 \end{bmatrix} d\mathbf{w}(t)$$

$g(x) := \exp(x)$,

$$Y_k = [1 \ 0 \ 0] X(t_k) + \xi_k, \quad \xi_k \sim N(0, 1),$$

The non-linear SDE above is a non-stationary extension of ~~the~~ a Matérn Gaussian process. It is useful in modelling non-stationary latent processes.

~~Now~~ ~~we apply the extended~~

Now generate a pair of $(X_{1:T}, Y_{1:T})$ then estimate $X_{1:T}$ by the extended Kalman filter and smoother. We need to take the Jacobian of the transition $\mathbb{E}[X(t_k) \mid X(t_{k-1}) = x]$.

See. 'lec6_extended_kf.ipynb' for how to do so.

Recall the general formulations of Gaussian filtering and smoothing in Algorithms 36 and 37, respectively. We see that ~~these filters and smoothers~~, the essence of them is really solving the Gaussian expectations. For instance, the idea of the extended Kalman filter is to linearise the integrand, so that we can solve the expectations in closed-form.

We can also the Gaussian expectations by quadrature, viz.,

$$\int g(x)\, \mathcal{N}(x \mid m, V)\, dx \approx \sum_{i=1}^{N} w_i\, g(x_i)$$

by a bunch of quadrature nodes and weights $\{x_i, w_i\}_{i=1}^{N}$. For example, we can let x_i's be the samples drawn from $\mathcal{N}(x \mid m, V)$. We can also choose them in terms of some orthonormal polynomials, for instance, Gauss-Hermite, unscented transform, and spherical cubature. ~~This~~ These gives the celebrated sigma-points filters and smoothers.

Gauss-Hermite quadrature (n-order) is exact for polynomial integrands of degree $\leq 2n-1$.