Intro to Autonomous Driving: Perception

Basic Algorithms in Sensor Fusion and SLAM

Jianan Liu

Gothenburg

August 20, 2018

Disclaimer

All opinions and statements in this presentation are mine and do not in any way represent the company
Any comment or correction of error is welcome, please contact me chisyliu@hotmail.com
All slides of sensor fusion are reproduced from Python Machine Learning Extra Book Circle I made years ago

Acknowledgement

Acknowledgement

Here I would like to thank all my colleagues from sensor fusion algorithm department and/or advanced safety department for their help in the daily work. I also would like to thank Mengbai Tao, Hui Wen and Ziqi Peng, for your friendship and help in making the original sensor fusion part of this slide. Good luck with your career in Combitech Gothenburg, Amazon and Goldman Sachs London.

Before We Start

In this presentation

- We will discuss two topics which are usually mentioned in autonomous driving, sensor fusion and SLAM
- I am sure this presentation will give you sufficient knowledge to understand what are sensor fusion and SLAM, the principle behind them, and in what ways(algorithms) to deal with it. Sensor fusion and SLAM are very important in the environment perception and autonomous driving
- If you want to know more details about sensor fusion, I recommend reading
- If you want to know more details about SLAM, visual SLAM, etc., I recommend reading

Before We Start

We actually have similar training samples for Bayesian filter and other machine learning algorithm(i.e. RLS for linear regression), but with little difference. We will discuss the details regarding the difference later, but for now we just define two kinds of training samples first

Data set $(\mathbf{y}, \mathbf{X})$ for RLS

Suppose we have L samples, each sample has D features/dimensions(So the input $\mathbf{X}$ is L by D matrix, label $\mathbf{y}$ is L by 1 vector). If we only consider the l^{th} data sample pair, $\mathbf{x}^{(l)}$ is 1 by D vector which represents D features/dimension for l^{th} training data sample, $\mathbf{w}$ is D by 1 vector which represents weight and $y^{(l)}$ represents the result of l^{th} data sample

Before We Start

Data series/sequence $(\mathbf{y}_t, \mathbf{x}_t)$ for Bayesian filter

Suppose we have T samples available in time series, each sample has D state variable $\mathbf{x}_t$ and observation $\mathbf{y}_t$ in time t

Note: Here we don't use word "data set" but "data series" to describe the training samples cause the data samples for Bayesian filter is available from time to time and the order of data matters the result

Overview

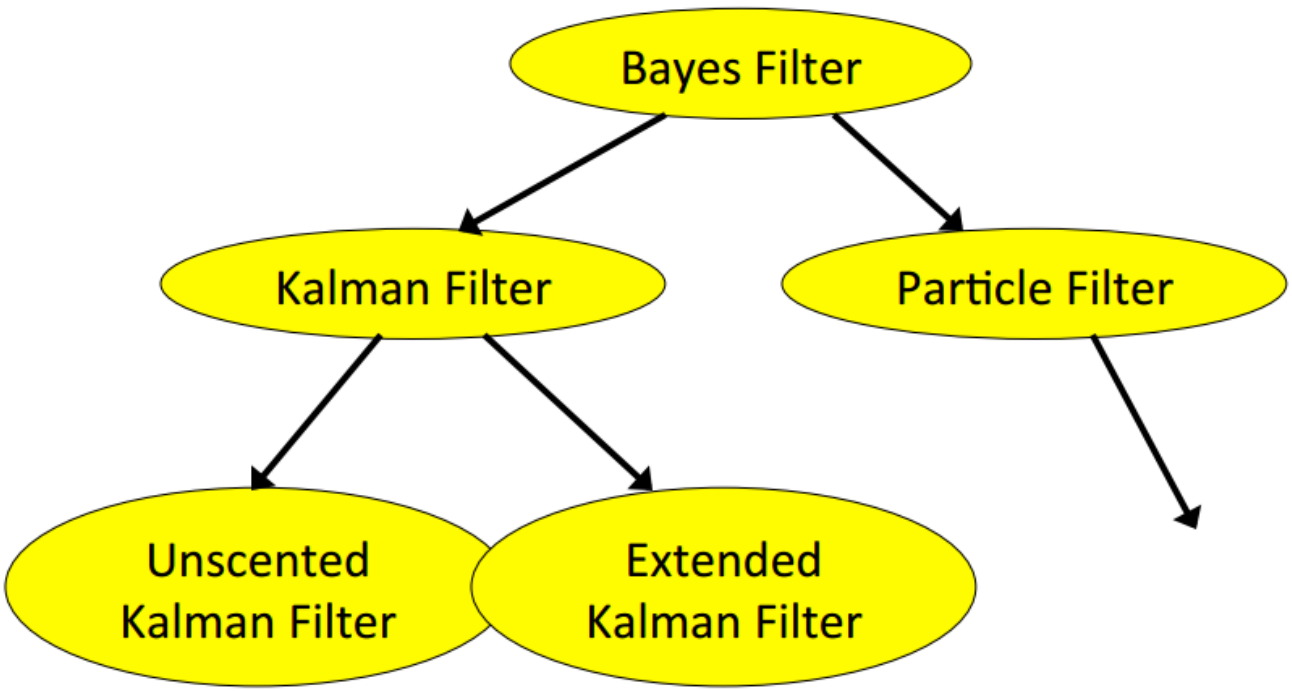
1. Part I: Introduction to Sensor Fusion
2. Part II: Bayesian Filter Family as Algorithms in Sensor Fusion
   - Bayesian Filter in General Format
   - Kalman Filter: A Specific Bayesian Filter for Linear Dynamic State System in Gaussian Distribution
   - Extended Kalman Filter: A Specific Bayesian Filter for Nonlinear Dynamic State System in Gaussian Distribution
   - Particle Filter: A Specific Bayesian Filter for Nonlinear Dynamic State System in NonGaussian Distribution
3. Part III: Introduction to SLAM
4. Part IV: Filter Based SLAM Algorithm
5. Part V: Keyframe Based/Optimization Based SLAM Algorithm

Outline for Section 1

Outline for Section 2

Bayesian Filter in General Format

- Figure from Ashutosh Saxena's course Robot Learning Course in Cornell University shows relationship between all Bayesian filters clearly



Bayesian Filter in General Format

Bayesian filter is actually used to solve the problem in SLAM: **state estimation**, or so called **localization**

- The state estimation is to calculate $P(\mathbf{x}|\mathbf{z})$ according to notations used for modeling SLAM problem, where x stands for state of a system, u is given of observations
- How to calculate this probability? By using Bayes rule $P(A|B) = \frac{P(B|A)P(A)}{P(B)} = \frac{P(A \bigcap B)}{P(B)}$ and assumption of independent (random variables) measurements(Markovian chain)

Bayesian Filter in General Format

- For the time stamp t, the estimation of state $\mathbf{x}_t$ could be expressed as $P(\mathbf{x}_t|\mathbf{y}_{1:t})$, where $\mathbf{y}_{1:t}$ stands for the sequence of observation from time stamp 1 to t
- Clearly this is a MAP rule, we try to check the probability of obtaining $\mathbf{x}_t$ based on availability of sequence of observation $\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_t$
- Now let's see what will be happened on the estimation of state $\mathbf{x}_t$: $P(\mathbf{x}_t|\mathbf{y}_{1:t})$ if applying Bayes rule and assumption of independent measurements
- By applying Bayes rule, $P(\mathbf{x}_t|\mathbf{y}_{1:t}) = P(\mathbf{y}_{1:t}|\mathbf{x}_t)P(\mathbf{x}_t)/P(\mathbf{y}_{1:t})$

Bayesian Filter in General Format

- If we assume all measurements $\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_t$ are conditional independent on state $\mathbf{x}_t$, and we also have $P(\mathbf{y}_{1:t}|\mathbf{x}_t) \equiv P((\mathbf{y}_{1:t-1}|\mathbf{x}_t) \bigcap (\mathbf{y}_t|\mathbf{x}_t)))$. Then we can get $P(\mathbf{y}_{1:t}|\mathbf{x}_t) \equiv P((\mathbf{y}_{1:t-1}|\mathbf{x}_t) \bigcap (\mathbf{y}_t|\mathbf{x}_t)) = P(\mathbf{y}_{1:t-1}|\mathbf{x}_t)P(\mathbf{y}_t|\mathbf{x}_t)$
- So we will have $P(\mathbf{x}_t|\mathbf{y}_{1:t}) = P(\mathbf{y}_{1:t}|\mathbf{x}_t)P(\mathbf{x}_t)/P(\mathbf{y}_{1:t}) = P(\mathbf{y}_{1:t-1}|\mathbf{x}_t)P(\mathbf{y}_t|\mathbf{x}_t)P(\mathbf{x}_t)/P(\mathbf{y}_{1:t})$
- Switch first and second part then we get $P(\mathbf{x}_t|\mathbf{y}_{1:t}) = P(\mathbf{y}_t|\mathbf{x}_t)P(\mathbf{y}_{1:t-1}|\mathbf{x}_t)P(\mathbf{x}_t)/P(\mathbf{y}_{1:t})$
- We apply Bayes rule once more on the second and third parts: $P(\mathbf{y}_{1:t-1}|\mathbf{x}_t)P(\mathbf{x}_t) = P(\mathbf{x}_t|\mathbf{y}_{1:t-1})P(\mathbf{y}_{1:t-1})$

Bayesian Filter in General Format

- Plug into the origin, we have $P(\mathbf{x}_t|\mathbf{y}_{1:t}) = P(\mathbf{y}_t|\mathbf{x}_t)P(\mathbf{x}_t|\mathbf{y}_{1:t-1})P(\mathbf{y}_{1:t-1})/P(\mathbf{y}_{1:t}) = \frac{P(\mathbf{y}_{1:t-1})}{P(\mathbf{y}_{1:t})}P(\mathbf{y}_t|\mathbf{x}_t)P(\mathbf{x}_t|\mathbf{y}_{1:t-1})$
- Until now, we have three parts in the equation which represents the Posterior of state estimation $P(\mathbf{x}_t|\mathbf{y}_{1:t})$:

- 1st part: $\frac{P(\mathbf{y}_{1:t-1})}{P(\mathbf{y}_{1:t})}$, which is a normalization constant cause $\mathbf{y}$ is observation(which is given value)
- 2nd part: $P(\mathbf{y}_t|\mathbf{x}_t)$, which clearly represents a likelihood function (consider a least square regression model $\mathbf{y}_t = \mathbf{x}_t\mathbf{w}_t$ in case $\mathbf{x}_t$ is the variable/weight and $\mathbf{w}_t$ is t^{th} training samples)
- 3rd part: $P(\mathbf{x}_t|\mathbf{y}_{1:t-1})$

Bayesian Filter in General Format

What is the third part? Could we rewrite it into other format, e.g. a format of $P(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1})$? If it could be written in that format, we could achieve a relationship between $P(\mathbf{x}_t|\mathbf{y}_{1:t})$ and $P(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1})$ in the recursive format for our original expression of state estimation $P(\mathbf{x}_t|\mathbf{y}_{1:t})$. So let's consider if it is possible

- First let's consider a joint probability $P((\mathbf{x}_t|\mathbf{y}_{1:t-1}) \bigcap P(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1}))$
- We just rewrite the joint probability in another way: $P((\mathbf{x}_t|\mathbf{y}_{1:t-1}) \bigcap P(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1})) \equiv P(\mathbf{x}_t, \mathbf{x}_{t-1}|\mathbf{y}_{1:t-1})$
- With setting $A = (\mathbf{x}_t|\mathbf{y}_{1:t-1}), B = (\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1})$, we have $P((\mathbf{x}_t|\mathbf{y}_{1:t-1}) \bigcap P(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1})) \equiv P(\mathbf{x}_t, \mathbf{x}_{t-1}|\mathbf{y}_{1:t-1}) = P(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{y}_{1:t-1})P(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1})$ cause $P(A \bigcap B) = P(A|B)P(B)$

Bayesian Filter in General Format

- Due $P(A) = \int P(A \bigcap B)\, dB = \int P(A|B)P(B)\, dB$, so we get $P(\mathbf{x}_t|\mathbf{y}_{1:t-1}) = \int P((\mathbf{x}_t|\mathbf{y}_{1:t-1}) \bigcap P(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1}))\, d\mathbf{x}_{t-1} = \int P(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{y}_{1:t-1})P(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1})\, d\mathbf{x}_{t-1}$ which is the **prediction step**
- Here the equation $\int P(A \bigcap B)\, dB = \int P(A|B)P(B)\, dB = P(A)$ is called law of total probability. (It is $\sum_B P(A|B)P(B)$ in discrete case)
- Now we plug the new expression of $P(\mathbf{x}_t|\mathbf{y}_{1:t-1})$ into third part on the original expression of $P(\mathbf{x}_t|\mathbf{y}_{1:t})$, we have $P(\mathbf{x}_t|\mathbf{y}_{1:t}) = \frac{P(\mathbf{y}_{1:t-1})}{P(\mathbf{y}_{1:t})} P(\mathbf{y}_t|\mathbf{x}_t)P(\mathbf{x}_t|\mathbf{y}_{1:t-1}) = \frac{P(\mathbf{y}_{1:t-1})}{P(\mathbf{y}_{1:t})} \cdot P(\mathbf{y}_t|\mathbf{x}_t) \cdot \int P(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{y}_{1:t-1})P(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1})\, d\mathbf{x}_{t-1}$ which is the **update step**

Bayesian Filter in General Format

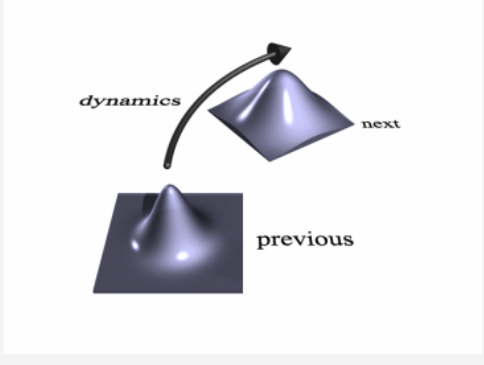
- By assumption of Markov procession of dynamic model $P(\mathbf{x}_t|\mathbf{x}_{t-1})$, which means we have $\mathbf{x}_t$ is only dependent on previous $\mathbf{x}_{t-1}$
- On the other word, $P(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{y}_{1:t-1}) = P(\mathbf{x}_t|\mathbf{x}_{t-1})$
- It means we finally fetch the connection between $P(\mathbf{x}_t|\mathbf{y}_{1:t})$ and $P(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1})$ in a recursive expression

Recursive Expression of Bayesian Filter in General Format

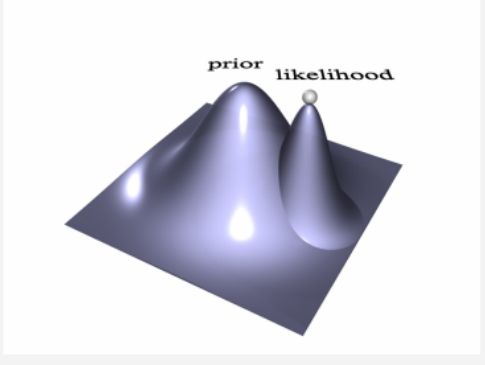
- $$P(\mathbf{x}_t|\mathbf{y}_{1:t}) = \frac{P(\mathbf{y}_{1:t-1})}{P(\mathbf{z}_{1:t})} P(\mathbf{y}_t|\mathbf{x}_t) P(\mathbf{x}_t|\mathbf{y}_{1:t-1}) = \frac{P(\mathbf{y}_{1:t-1})}{P(\mathbf{y}_{1:t})} \cdot P(\mathbf{y}_t|\mathbf{x}_t) \cdot \int P(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{y}_{1:t-1}) P(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1}) \, d\mathbf{x}_{t-1} = \underbrace{\frac{P(\mathbf{y}_{1:t-1})}{P(\mathbf{y}_{1:t})} \cdot P(\mathbf{y}_t|\mathbf{x}_t) \cdot \underbrace{\int P(\mathbf{x}_t|\mathbf{x}_{t-1}) P(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1}) \, d\mathbf{x}_{t-1}}_{\textit{prediction}}}_{\textit{update}}$$

Graphic Explanation of Distribution Movement from Prediction to Update in Bayesian Filter

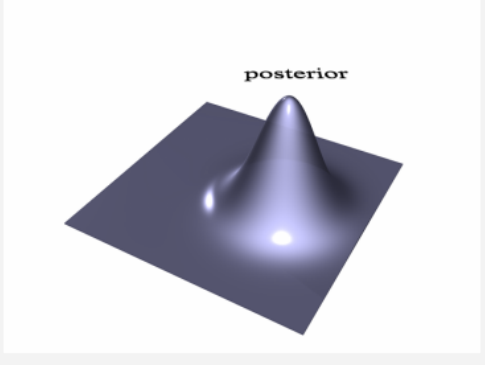
Now let's borrow the figure from Simo Sarkkas course Nonlinear Filtering and Estimation in Aalto University, to show how the distribution moves from prediction part to update part in Bayesian filter in general format



On prediction step the distribution of previous step is propagated through the dynamics.



Prior distribution from prediction and the likelihood of measurement.



The posterior distribution after combining the prior and likelihood by Bayes' rule.

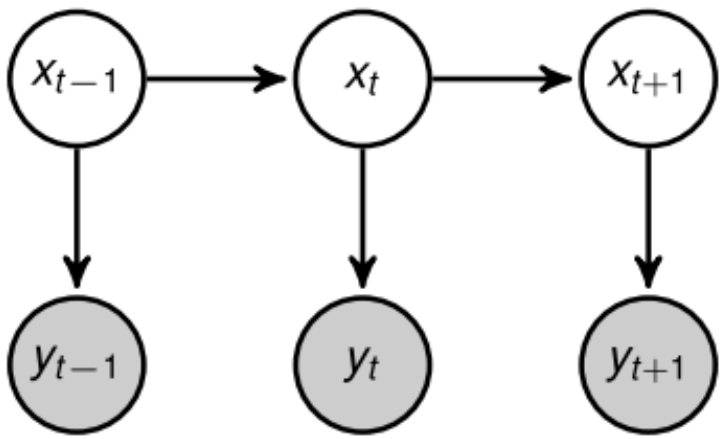
Graphic Explanation of Distribution Movement from Prediction to Update in Bayesian Filter

- Clearly the distribution of previous step $P(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1})$ has been propagated on prediction step
- In the update step, the distribution of likelihood measurement $P(\mathbf{y}_t|\mathbf{x}_t)$ has been combined with prior distribution from prediction $\int P(\mathbf{x}_t|\mathbf{x}_{t-1})P(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1})\, d\mathbf{x}_{t-1}$
- According to Bayes rule, we actually get the posterior distribution $P(\mathbf{x}_t|\mathbf{y}_{1:t})$ after combining the prior and likelihood

Behind Bayesian Filter: Hidden Markov Model

The Bayesian filter could be explained by using hidden markov modle(HMM) as well

- Let's see an example of HMM in figure, where the hidden state $\mathbf{x}_t$ are only dependent on previous state $\mathbf{x}_{t-1}$, but the observation $\mathbf{y}_t$ are independent on all other observations and the observation $\mathbf{y}_t$ are conditioned on the hidden state $\mathbf{x}_t$



Behind Bayesian Filter: Hidden Markov Model

The Bayesian filter could be explained by using hidden markov modle(HMM) as well

- Clearly, our needed assumptions for Bayesian filter are same as the characteristics belongs to HMM. Which are:
  1. All observations $\mathbf{y}_t$ are conditional independent on state $\mathbf{x}_t$
  2. The state $\mathbf{x}_t$ is a Markov process
- It means the HMM could describe Bayesian filter for dynamic system very well

From Bayesian Filter to Kalman Filter

We already understood how a Bayesian filter works in a recursive expression which established relationship between $P(\mathbf{x}_t|\mathbf{y}_{1:t})$ and $P(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1})$. But that is just a generic format without specifying what is the exact relationship(a specific mathematical expression) between hidden state $\mathbf{x}_t$ and $\mathbf{x}_{t-1}$, and the exact relationship between hidden state $\mathbf{x}_t$ and observation/measurements $\mathbf{y}_t$. So now the question comes, how does Bayesian filter work for a exact real system (model)?

- From this sub section, we will introduce the concept of Gaussian linear dynamic state system and see how a Bayesian filter in general evolves to Kalman filter in this specific system model

From Bayesian Filter to Kalman Filter

First let's check what is Gaussian linear dynamic state system

- A Gaussian linear dynamic state system could be described by two equations:
  1. $\mathbf{x}_t = \mathbf{F}_{t-1}\mathbf{x}_{t-1} + \mathbf{q}_{t-1}$
  2. $\mathbf{y}_t = \mathbf{H}_t\mathbf{x}_t + \mathbf{r}_t$
- For 1st equation, where $\mathbf{x}_t$ stands for the system state(which is a variable we do NOT know, and the task is to estimate the value of $\mathbf{x}_t$), $\mathbf{F}_{t-1}$ stands for the transition of state from t-1 to t, $\mathbf{q}_{t-1}$ is a Gaussian noise with zero mean and covariance $\mathbf{Q}_{t-1}$
- For 2nd equation, where $\mathbf{y}_t$ stands for the observation/measurement, $\mathbf{H}_t$ stands for the measurement loss of $\mathbf{x}_t$, $\mathbf{r}_t$ is a Gaussian noise with zero mean and covariance $\mathbf{R}_t$

From Bayesian Filter to Kalman Filter

- Notice $\mathbf{x}_t$ follows Gaussian distribution by the first equation. And $\mathbf{y}_t$ follows Gaussian as well cause second equation and $\mathbf{x}_t$ is Gaussian

In short, Gaussian linear dynamic state system looks as below

Gaussian Linear Dynamic State System Model

- $\mathbf{x}_t = \mathbf{F}_{t-1}\mathbf{x}_{t-1} + \mathbf{q}_{t-1}$ (This is called "motion model")
- $\mathbf{y}_t = \mathbf{H}_t\mathbf{x}_t + \mathbf{r}_t$ (This is called "observation model")
- $\mathbf{q}_{t-1} \sim \mathcal{N}(0, \mathbf{Q}_{t-1})$
- $\mathbf{r}_t \sim \mathcal{N}(0, \mathbf{R}_t)$

From Bayesian Filter to Kalman Filter

From now we start to see how Bayesian filter evolves Kalman filter for Gaussian Linear Dynamic State System Model, and more importantly, how Kalman filter works

- Firstly, we could represent the state $\mathbf{x}_t$ based on observations as Gaussian distribution as below:
- $P(\mathbf{x}_t|\mathbf{y}_{1:t}) = \mathcal{N}(\mathbf{m}_t, \mathbf{P}_t)$ which denotes the probability of state $\mathbf{x}_t$ based on observations from 1 to t
- $P(\mathbf{x}_t|\mathbf{y}_{1:t-1}) = \mathcal{N}(\mathbf{m}_t^-, \mathbf{P}_t^-)$ which denotes the probability of state $\mathbf{x}_t$ based on observations from 1 to t-1(lack of observation $\mathbf{y}_t$, on the other word, it is the estimation of state $\mathbf{x}_t$ before acquiring observation $\mathbf{y}_t$)

From Bayesian Filter to Kalman Filter

Now let's introduce a lemma(it is correct and we could use it)

Lemma 1

- if two variables $\mathbf{a}$ and $\mathbf{b}$ have:
  $P(\mathbf{a}) = \mathcal{N}(\mathbf{a}|\mathbf{m}, \mathbf{P})$ (variable $\mathbf{a}$ follows Gaussian with mean $\mathbf{m}$ and covariance $\mathbf{P}$) and
  $P(\mathbf{b}|\mathbf{a}) = \mathcal{N}(\mathbf{b}|\mathbf{Ha}, \mathbf{R})$ (variable $\mathbf{b}$'s mean has the relationship $\mathbf{Ha}$ with variable $\mathbf{a}$ and covariance is $\mathbf{R}$), actually means
  $\mathbf{b} = \mathbf{Ha} + n$ where n is a Gaussian variable follows $\mathcal{N}(0, \mathbf{R})$)
- then the joint distribution of $\mathbf{a}$ and $\mathbf{b}$ is:
  $$\mathbf{a} \bigcap \mathbf{b} = \begin{pmatrix} \mathbf{a} \\ \mathbf{b} \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} \mathbf{m} \\ \mathbf{Hm} \end{pmatrix}, \begin{pmatrix} \mathbf{P} & \mathbf{PH}^T \\ \mathbf{HP} & \mathbf{HPH}^T + \mathbf{R} \end{pmatrix} \right)$$
- and marginal distribution of $\mathbf{b}$ is:
  $\mathbf{b} \sim \mathcal{N}(\mathbf{Hm}, \mathbf{HPH}^T + \mathbf{R})$

From Bayesian Filter to Kalman Filter

- From the two conditional probabilities we defined on the page before previous one, we actual could have:
  $P(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1}) = \mathcal{N}(\mathbf{m}_{t-1}, \mathbf{P}_{t-1})$
  $P(\mathbf{x}_t|\mathbf{y}_{1:t-1}) = \mathcal{N}(\mathbf{m}_t^-, \mathbf{P}_t^-)$
- Recall our prediction step for Bayesian filter:
  $P(\mathbf{x}_t|\mathbf{y}_{1:t-1}) = \int P((\mathbf{x}_t|\mathbf{y}_{1:t-1}) \bigcap P(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1}))\, d\mathbf{x}_{t-1} =$
  $\int P(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{y}_{1:t-1}) P(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1})\, d\mathbf{x}_{t-1}$
  Here inside the integration we actually have a joint probability of "$\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1}$" and "$\mathbf{x}_t|\mathbf{y}_{1:t-1}$" in prediction step and we actually want to represent the prediction
  $P(\mathbf{x}_t|\mathbf{y}_{1:t-1}) = \mathcal{N}(\mathbf{m}_t^-, \mathbf{P}_t^-)$ by using
  $P(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1}) = \mathcal{N}(\mathbf{m}_{t-1}, \mathbf{P}_{t-1})$

From Bayesian Filter to Kalman Filter

- Notice $\mathbf{x}_t = \mathbf{F}_{t-1}\mathbf{x}_{t-1} + \mathbf{q}_{t-1}$, which satisfies the requirement between two Gaussian variables in lemma 1.
- So we could apply lemma 1 on $\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1}$ and $\mathbf{x}_t|\mathbf{y}_{1:t-1}$ by define $\mathbf{a} = \mathbf{x}_{t-1}|\mathbf{y}_{1:t-1}$ and $\mathbf{b} = \mathbf{x}_t|\mathbf{y}_{1:t-1}$
- On the other word, we have $P(\mathbf{a}) = P(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1}) = \mathcal{N}(\mathbf{m}_{t-1}, \mathbf{P}_{t-1})$
  and $P(\mathbf{b}|\mathbf{a}) = P(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{y}_{1:t-1}) = P(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t|\mathbf{F}_{t-1}\mathbf{x}_{t-1}, \mathbf{Q}_{t-1})$ cause of $\mathbf{x}_t$ is Markov process and $\mathbf{x}_t = \mathbf{F}_{t-1}\mathbf{x}_{t-1} + \mathbf{q}_{t-1}$. It means $\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1}$ and $\mathbf{x}_t|\mathbf{y}_{1:t-1}$ satisfy the "if" requirement in lemma 1

From Bayesian Filter to Kalman Filter

- So we get the joint probability:
  $P(\mathbf{a} \bigcap \mathbf{b}) = P((\mathbf{x}_t|\mathbf{y}_{1:t-1}) \bigcap P(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1})) =$
  $P((\mathbf{x}_t, \mathbf{x}_{t-1}|\mathbf{y}_{1:t-1}))=$
  $$\mathcal{N}\left(\begin{pmatrix} \mathbf{m}_{t-1} \\ \mathbf{F}_{t-1}\mathbf{m}_{t-1} \end{pmatrix}, \begin{pmatrix} \mathbf{P}_{t-1} & \mathbf{P}_{t-1}\mathbf{F}_{t-1}^T \\ \mathbf{F}_{t-1}\mathbf{P}_{t-1} & \mathbf{F}_{t-1}\mathbf{P}_{t-1}\mathbf{F}_{t-1}^T + \mathbf{Q}_{t-1} \end{pmatrix}\right)$$
- also marginal distribution of $\mathbf{b} = \mathbf{x}_t|\mathbf{y}_{1:t-1}$:
  $P(\mathbf{b}) = P(\mathbf{x}_t|\mathbf{y}_{1:t-1}) = \mathcal{N}(\mathbf{F}_{t-1}\mathbf{m}_{t-1}, \mathbf{F}_{t-1}\mathbf{P}_{t-1}\mathbf{F}_{t-1}^T + \mathbf{Q}_{t-1})$
- cause we also defined $P(\mathbf{x}_t|\mathbf{y}_{1:t-1}) = \mathcal{N}(\mathbf{m}_t^-, \mathbf{P}_t^-)$, then we can use the mean and covariance of $\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1}$, $\mathbf{m}_{t-1}$ and $\mathbf{P}_{t-1}$, to represent the mean and covariance of $\mathbf{x}_t|\mathbf{y}_{1:t-1}$, $\mathbf{m}_t^-$ and $\mathbf{P}_t^-$

From Bayesian Filter to Kalman Filter

By getting $P(\mathbf{x}_t|\mathbf{y}_{1:t-1}) = \mathcal{N}(\mathbf{m}_t^-, \mathbf{P}_t^-) = \mathcal{N}(\mathbf{F}_{t-1}\mathbf{m}_{t-1}, \mathbf{F}_{t-1}\mathbf{P}_{t-1}\mathbf{F}_{t-1}^T + \mathbf{Q}_{t-1})$:

- $\mathbf{m}_t^- = \mathbf{F}_{t-1}\mathbf{m}_{t-1}$
- and $\mathbf{P}_t^- = \mathbf{F}_{t-1}\mathbf{P}_{t-1}\mathbf{F}_{t-1}^T + \mathbf{Q}_{t-1}$
- Notice $P(\mathbf{x}_t|\mathbf{y}_{1:t-1})$ stands for the prediction step of Bayesian filter, and now we are in Gaussian linear dynamic system model, so here we actually get the prediction step in Kalman filter

Prediction Step of Kalman Filter

- $\mathbf{m}_t^- = \mathbf{F}_{t-1}\mathbf{m}_{t-1}$
- $\mathbf{P}_t^- = \mathbf{F}_{t-1}\mathbf{P}_{t-1}\mathbf{F}_{t-1}^T + \mathbf{Q}_{t-1}$

From Bayesian Filter to Kalman Filter

- Now we already got $P(\mathbf{x}_t|\mathbf{y}_{1:t-1})$ in the expression of $P(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1})$ from prediction step, we move to consider the update step $P(\mathbf{x}_t|\mathbf{z}_{1:t}) = \frac{P(\mathbf{y}_{1:t-1})}{P(\mathbf{y}_{1:t})} P(\mathbf{y}_t|\mathbf{x}_t) P(\mathbf{x}_t|\mathbf{y}_{1:t-1})$
- Our purpose is to find the relationship between $P(\mathbf{x}_t|\mathbf{z}_{1:t})$ and $P(\mathbf{x}_{t-1}|\mathbf{z}_{1:t-1})$ through $P(\mathbf{x}_t|\mathbf{y}_{1:t-1})$
- This time define $\mathbf{a} = \mathbf{x}_t|\mathbf{y}_{1:t-1}$ and $\mathbf{b} = \mathbf{y}_t|\mathbf{y}_{1:t-1}$
- We notice $P(\mathbf{y}_t|\mathbf{x}_t)P(\mathbf{x}_t|\mathbf{y}_{1:t-1}) = P(\mathbf{y}_t|\mathbf{x}_t, \mathbf{y}_{1:t-1})P(\mathbf{x}_t|\mathbf{y}_{1:t-1}) = P(\mathbf{b}|\mathbf{a})P(\mathbf{a}) = P(\mathbf{x}_t, \mathbf{y}_t|\mathbf{y}_{1:t-1}) = P(\mathbf{a} \bigcap \mathbf{b})$ cause $P(\mathbf{y}_t|\mathbf{x}_t$ is independent on $P(\mathbf{y}_{1:t-1})$

From Bayesian Filter to Kalman Filter

- And we also notice $\mathbf{y}_t = \mathbf{H}_t\mathbf{x}_t + \mathbf{r}_t$
- So we could get $P(\mathbf{b}|\mathbf{a}) = P(\mathbf{y}_t|\mathbf{x}_t, \mathbf{y}_{1:t-1}) = P(\mathbf{y}_t|\mathbf{x}_t) = \mathcal{N}(\mathbf{y}_t|\mathbf{H}_t\mathbf{x}_t, \mathbf{R}_t)$
- Previously we defined $P(\mathbf{x}_t|\mathbf{y}_{1:t-1}) = \mathcal{N}(\mathbf{m}_t^-, \mathbf{P}_t^-)$, so we have $P(\mathbf{a}) = P(\mathbf{x}_t|\mathbf{y}_{1:t-1}) = \mathcal{N}(\mathbf{m}_t^-, \mathbf{P}_t^-)$
- By applying lemma 1 on variable $\mathbf{a} = \mathbf{x}_t|\mathbf{y}_{1:t-1}$ and $\mathbf{b} = \mathbf{y}_t|\mathbf{y}_{1:t-1}$, we could get the joint distribution of $\mathbf{a}$ and $\mathbf{b}$ $P((\mathbf{x}_t, \mathbf{y}_t|\mathbf{y}_{1:t-1}))$ on next page:

From Bayesian Filter to Kalman Filter

- The joint probability:
  $P(\mathbf{a} \bigcap \mathbf{b}) = P((\mathbf{x}_t|\mathbf{y}_{1:t-1}) \bigcap P(\mathbf{y}_t|\mathbf{y}_{1:t-1})) =$
  $P((\mathbf{x}_t, \mathbf{y}_t|\mathbf{y}_{1:t-1}))=$
  $$\mathcal{N}\left(\begin{pmatrix} \mathbf{m}_t^- \\ \mathbf{H}_t\mathbf{m}_t^- \end{pmatrix}, \begin{pmatrix} \mathbf{P}_t^- & \mathbf{P}_t^-\mathbf{H}_t^T \\ \mathbf{H}_t\mathbf{P}_t^- & \mathbf{H}_t\mathbf{P}_t^-\mathbf{H}_t^T + \mathbf{R}_t \end{pmatrix}\right)$$
- also marginal distribution of $\mathbf{b} = \mathbf{y}_t|\mathbf{y}_{1:t-1}$:
  $P(\mathbf{b}) = P(\mathbf{y}_t|\mathbf{y}_{1:t-1}) = \mathcal{N}(\mathbf{H}_t\mathbf{m}_t^-, \mathbf{H}_t\mathbf{P}_t^-\mathbf{H}_t^T + \mathbf{R}_t)$

From Bayesian Filter to Kalman Filter

Now let's introduce second lemma

Lemma 2

- if we have two variables $\mathbf{a}$ and $\mathbf{b}$ and the joint distribution of $\mathbf{a}$ and $\mathbf{b}$ is:
$$\mathbf{a} \bigcap \mathbf{b} = \begin{pmatrix} \mathbf{a} \\ \mathbf{b} \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} \alpha \\ \beta \end{pmatrix}, \begin{pmatrix} \mathbf{X} & \mathbf{Z}^T \\ \mathbf{Z} & \mathbf{Y} \end{pmatrix} \right)$$
- then the marginal and conditional densities are:
$$P(\mathbf{a}) = \mathcal{N}(\alpha, \mathbf{X})$$
$$P(\mathbf{b}) = \mathcal{N}(\beta, \mathbf{Y})$$
$$P(\mathbf{a}|\mathbf{b}) = \mathcal{N}(\alpha + \mathbf{Z}^T\mathbf{Y}^{-1}(\mathbf{b} - \beta), \mathbf{x} - \mathbf{Z}^T\mathbf{Y}^{-1}\mathbf{Z})$$
$$P(\mathbf{b}|\mathbf{a}) = \mathcal{N}(\beta + \mathbf{Z}\mathbf{X}^{-1}(\mathbf{a} - \alpha), \mathbf{Y} - \mathbf{Z}\mathbf{X}^{-1}\mathbf{Z}^T)$$

From Bayesian Filter to Kalman Filter

- By applying lemma 2 on the joint probability of $\mathbf{a}$ and $\mathbf{b}$ we got on the page before previous one, $P(\mathbf{a} \bigcap \mathbf{b}) = P((\mathbf{x}_t|\mathbf{y}_{1:t-1}) \bigcap P(\mathbf{y}_t|\mathbf{y}_{1:t-1})) = P((\mathbf{x}_t, \mathbf{y}_t|\mathbf{y}_{1:t-1})) = \mathcal{N}\left(\begin{pmatrix} \mathbf{m}_t^- \\ \mathbf{H}_t\mathbf{m}_t^- \end{pmatrix}, \begin{pmatrix} \mathbf{P}_t^- & \mathbf{P}_t^-\mathbf{H}_t^T \\ \mathbf{H}_t\mathbf{P}_t^- & \mathbf{H}_t\mathbf{P}_t^-\mathbf{H}_t^T + \mathbf{R}_t \end{pmatrix}\right)$
- We notice $P(\mathbf{a}|\mathbf{b}) = P(\mathbf{x}_t|\mathbf{y}_{1:t-1}, \mathbf{y}_t) = P(\mathbf{x}_t|\mathbf{y}_{1:t})$ is the final variable we want to calculate from update step, and we already defined $P(\mathbf{x}_t|\mathbf{y}_{1:t}) = \mathcal{N}(\mathbf{m}_t, \mathbf{P}_t)$
- We could get the conditional density function $P(\mathbf{a}|\mathbf{b}) = P(\mathbf{x}_t|\mathbf{y}_{1:t-1}, \mathbf{y}_t) = P(\mathbf{x}_t|\mathbf{y}_{1:t}) = \mathcal{N}(\mathbf{m}_t, \mathbf{P}_t) = \mathcal{N}(\mathbf{m}_t^- + \mathbf{P}_t^-\mathbf{H}_t^T(\mathbf{H}_t\mathbf{P}_t^-\mathbf{H}_t^T + \mathbf{R}_t)^{-1}(\mathbf{y}_t - \mathbf{H}_t\mathbf{m}_t^-), \mathbf{P}_t^- - \mathbf{P}_t^-\mathbf{H}_t^T(\mathbf{H}_t\mathbf{P}_t^-\mathbf{H}_t^T + \mathbf{R}_t)^{-1}\mathbf{H}_t\mathbf{P}_t^-)$

From Bayesian Filter to Kalman Filter

- So it means:
  $\mathbf{m}_t = \mathbf{m}_t^- + \mathbf{P}_t^- \mathbf{H}_t^T (\mathbf{H}_t \mathbf{P}_t^- \mathbf{H}_t^T + \mathbf{R}_t)^{-1} (\mathbf{y}_t - \mathbf{H}_t \mathbf{m}_t^-)$
  and $\mathbf{P}_t = \mathbf{P}_t^- - \mathbf{P}_t^- \mathbf{H}_t^T (\mathbf{H}_t \mathbf{P}_t^- \mathbf{H}_t^T + \mathbf{R}_t)^{-1} \mathbf{H}_t \mathbf{P}_t^-$
- We use new variables to make the equations simpler:
  Let $\mathbf{S}_t = \mathbf{H}_t \mathbf{P}_t^- \mathbf{H}_t^T + \mathbf{R}_t$
  $\mathbf{v}_t = \mathbf{y}_t - \mathbf{H}_t \mathbf{m}_t^-$
  So we have $\mathbf{K}_t = \mathbf{P}_t^- \mathbf{H}_t^T (\mathbf{H}_t \mathbf{P}_t^- \mathbf{H}_t^T + \mathbf{R}_t)^{-1} = \mathbf{P}_t^- \mathbf{H}_t^T \mathbf{S}_t^{-1}$
- Then we rewrite the two equations, we get:
  $\mathbf{m}_t = \mathbf{m}_t^- + \mathbf{K}_t \mathbf{v}_t$
  $\mathbf{P}_t = \mathbf{P}_t^- - \mathbf{K}_t \mathbf{S}_t \mathbf{K}_t^T$

From Bayesian Filter to Kalman Filter

So we actually rewrite what we got together, it is from update step of Bayesian filter with applying for Gaussian linear dynamic system model. So we actually get update step of Kalman filter

Measurements Update Step of Kalman Filter

- $\mathbf{S}_t = \mathbf{H}_t \mathbf{P}_t^- \mathbf{H}_t^T + \mathbf{R}_t$
- $\mathbf{v}_t = \mathbf{y}_t - \mathbf{H}_t \mathbf{m}_t^-$
- $\mathbf{K}_t = \mathbf{P}_t^- \mathbf{H}_t^T (\mathbf{H}_t \mathbf{P}_t^- \mathbf{H}_t^T + \mathbf{R}_t)^{-1} = \mathbf{P}_t^- \mathbf{H}_t^T \mathbf{S}_t^{-1}$
- $\mathbf{m}_t = \mathbf{m}_t^- + \mathbf{K}_t \mathbf{v}_t$
- $\mathbf{P}_t = \mathbf{P}_t^- - \mathbf{K}_t \mathbf{S}_t \mathbf{K}_t^T$

where $\mathbf{K}_t$ is Kalman gain

From Bayesian Filter to Kalman Filter

Combine prediction and update, we get Kalman filter($\mathbf{m}_t = \mathbf{x}_t$ for random variable with Gaussian distribution)

Predication Step of Kalman Filter

- $\mathbf{m}_t^- = \mathbf{F}_{t-1}\mathbf{m}_{t-1}$
- $\mathbf{P}_t^- = \mathbf{F}_{t-1}\mathbf{P}_{t-1}\mathbf{F}_{t-1}^T + \mathbf{Q}_{t-1}$

Measurements Update Step of Kalman Filter

- $\mathbf{S}_t = \mathbf{H}_t\mathbf{P}_t^-\mathbf{H}_t^T + \mathbf{R}_t$
- $\mathbf{v}_t = \mathbf{y}_t - \mathbf{H}_t\mathbf{m}_t^-$
- $\mathbf{K}_t = \mathbf{P}_t^-\mathbf{H}_t^T(\mathbf{H}_t\mathbf{P}_t^-\mathbf{H}_t^T + \mathbf{R}_t)^{-1} = \mathbf{P}_t^-\mathbf{H}_t^T\mathbf{S}_t^{-1}$
- $\mathbf{m}_t = \mathbf{m}_t^- + \mathbf{K}_t\mathbf{v}_t$
- $\mathbf{P}_t = \mathbf{P}_t^- - \mathbf{K}_t\mathbf{S}_t\mathbf{K}_t^T$

Kalman Filter Algorithm(Start from t = 1)

for t in T **do**
    // Prediction step of Kalman Filter
    $\mathbf{x}_t^- = \mathbf{F}_{t-1}\mathbf{x}_{t-1}$
    $\mathbf{P}_t^- = \mathbf{F}_{t-1}\mathbf{P}_{t-1}\mathbf{F}_{t-1}^T + \mathbf{Q}_{t-1}$
    // Measurements Update Step of Kalman Filter
    $\mathbf{S}_t = \mathbf{H}_t\mathbf{P}_t^-\mathbf{H}_t^T + \mathbf{R}_t$
    $\mathbf{v}_t = \mathbf{y}_t - \mathbf{H}_t\mathbf{x}_t^-$
    $\mathbf{K}_t = \mathbf{P}_t^-\mathbf{H}_t^T\mathbf{S}_t^{-1}$
    $\mathbf{x}_t = \mathbf{x}_t^- + \mathbf{K}_t\mathbf{v}_t$
    $\mathbf{P}_t = \mathbf{P}_t^- - \mathbf{K}_t\mathbf{S}_t\mathbf{K}_t^T$
    t = t + 1
end for

Kalman Filter: As Optimization Problem of Maximize A Posterior(MAP)

- Previously in the first section "Bayesian Filter in General and Kalman Filter", we derivate Kalman filter analytic solution from Bayesian filter in conceptual/general by using two lemmas regarding the Gaussian distribution in following of Bayesian inference. In this sub section, we treat Kalman filter from the angel of a optimization problem which maximizes posterior in Gaussian
- But we only show the ideas and procedures rather than giving the exact detail of derivation, cause the purpose of showing this is just giving the other way of understanding Kalman filter. Refer to the chapter 3 of Timothy D. Barfoot's book State Estimation for Robotics if you are interested in detail how to derivate Kalman filter via MAP

Kalman Filter: As Optimization Problem of Maximize A Posterior(MAP)

From this page, we show the basic idea of how to get Kalman filter via MAP optimization problem

- First we actually want to do is to find out the estimation of $\mathbf{x}$ which could maximize the posterior $P(\mathbf{x}|\mathbf{y})$
- In the other words, $\widehat{\mathbf{x}} = argmax_{\mathbf{x}} P(\mathbf{x}|\mathbf{y})$
- Due to Bayes rule we have $P(\mathbf{x}|\mathbf{y}) = P(\mathbf{y}|\mathbf{x})P(\mathbf{x})/P(\mathbf{y})$
- So we actually want to $\widehat{\mathbf{x}} = argmax_{\mathbf{x}} P(\mathbf{y}|\mathbf{x})P(\mathbf{x})/P(\mathbf{y}) = argmax_{\mathbf{x}} P(\mathbf{y}|\mathbf{x})P(\mathbf{x})$

Kalman Filter: As Optimization Problem of Maximize A Posterior(MAP)

- Due to HMM, we know
  $P(\mathbf{y}|\mathbf{x}) = \Pi_{t=1}^{T} P(\mathbf{y}_t|\mathbf{x}_t)$
  and $P(\mathbf{x}) = P(\mathbf{x}_0|\widehat{\mathbf{x}}_0)\Pi_{t=1}^{T} P(\mathbf{x}_t|\mathbf{x}_{t-1})$
- $P(\mathbf{x}_0|\widehat{\mathbf{x}}_0)$, $P(\mathbf{x}_t|\mathbf{x}_{t-1})$, $P(\mathbf{y}_t|\mathbf{x}_t)$ are all Gaussian distribution. We could use the definition of Gaussian distribution density function to express them mathematically by using $\mathbf{F}_{t-1}$, $\mathbf{P}_t$, $\mathbf{Q}_t$, $\mathbf{R}_t$, etc.
- It means we could get the mathematical expression of $P(\mathbf{x}_0|\widehat{\mathbf{x}}_0)$, $P(\mathbf{x}_t|\mathbf{x}_{t-1})$, $P(\mathbf{y}_t|\mathbf{x}_t)$ exactly

Kalman Filter: As Optimization Problem of Maximize A Posterior(MAP)

- Then we go back to the original optimization problem, we take the ln on the objective function, so we have $\widehat{\mathbf{x}} = argmax_{\mathbf{x}}\ ln(P(\mathbf{y}|\mathbf{x})P(\mathbf{x})) = argmax_{\mathbf{x}}\ ln(\Pi_{t=1}^{T} P(\mathbf{y}_t|\mathbf{x}_t)) + ln(P(\mathbf{x}_0|\widehat{\mathbf{x}}_0)) + ln(\Pi_{t=1}^{T} P(\mathbf{x}_t|\mathbf{x}_{t-1}))$
- Then we take the inverse of maximize of posterior, so we get $\widehat{\mathbf{x}} = argmin_{\mathbf{x}}\ -ln(\Pi_{t=1}^{T} P(\mathbf{y}_t|\mathbf{x}_t)) - ln(P(\mathbf{x}_0|\widehat{\mathbf{x}}_0)) - ln(\Pi_{t=1}^{T} P(\mathbf{x}_t|\mathbf{x}_{t-1}))$
- This is a convex function, we set the derivative of objective which equal to 0 and calculate the analytical result of $\mathbf{x}$
- Then by manipulating a bit of result, we will get the same format as Kalman filter from Bayesian inference by using 2 Gaussian lemma

Kalman Filter: As Optimization Problem of Maximize A Posterior(MAP)

- Again, we do NOT derivate detail of this procedure via MAP step by step, but all detail could be found on chapter 3 of Timothy D. Barfoot's book State Estimation for Robotic
- So far, we know Kalman filter could be derivated either from Bayesian inference by using two Gaussian lemma, or from optimization problem maximize posterior solved analytically by using "inverse of ln" and definition of Gaussian distribution density function

Revisit Least Square Estimation: LMS/Gradient Descent for Machine Learning Problem

Firstly let us recall linear regression model $\mathbf{y} = \mathbf{X}\mathbf{w}$ which we mentioned in section 3 of 1st presentation Python Machine Learning book circle, there we introduced LMS

- LMS is a batch gradient descent based searching algorithm which searches in the direction of inversion of gradient iteratively.
- LMS uses 'average' of correlation of observed data samples to replace of 'expectation' of correlation of ensemble data set, which leads to a LS/ML sense approximation of MMSE analytical solution
- LMS needs to take all observed data samples into consideration for each iteration, which makes LMS NOT suitable for online/sequential learning/estimation

Revisit Least Square Estimation: LMS/Gradient Descent for Machine Learning Problem

- More importantly, LMS/Gradient Descent searching algorithm actually works for static system where the variable/weight $\mathbf{w}$ is static(NOT changing depends on time stamp t). If the variable/weight $\mathbf{w}$ is changing from time to time(e.g. $\mathbf{w}_t \neq \mathbf{w}_{t-1}$, it is dynamic state system), the $\mathbf{w}$ is a time-series where the order of data is important(you can NOT change the order of data samples for different time cause the trend of data carries information). LMS/Gradient Descent searching algorithm does NOT work for dynamic state system(It is why LMS/Gradient Descent searching algorithm, as algorithm for LS/ML criteria only works least square regression kind problems because the linear regression model is a static model where variable $\mathbf{w}$ NOT related to time)

Revisit Least Square Estimation: LMS/Gradient Descent for Machine Learning Problem

LMS/Gradient Descent does NOT fit for state estimation

- So LMS/Gradient Descent usually only works for machine learning problem, statistical signal processing and wireless communication, e.g.(which are static system and the variable does NOT change from time to time) rather than tracking, time-series analysis, e.g.(which are dynamic system and the variable varies on time) because the order of training data does NOT matter(training data does NOT depend on time) and the model in machine learning is usually NOT dynamic model(the variables/weights does NOT vary in different t/NOT related to t). All in all, LMS/Gradient Descent does NOT fit for (dynamic system)state estimation problem

Revisit Least Square Estimation: Recursive Least Square for Online/Sequential Estimation

Now recall section 4 of 1st presentation in Python Machine Learning book circle, an alternative solution RLS was introduced for the same linear model $\mathbf{y} = \mathbf{X}\mathbf{w}$

- RLS is NOT gradient descent based searching algorithm
- Instead of searching in the direction of inversion of gradient iteratively, **RLS actually calculate the $\mathbf{w}_{LS}(l)$ of current l samples for l^{th} recursive step**
- Calculation of $\mathbf{w}_{LS}(l)$ for each recursive step is not directly from analytic solution but using matrix inversion lemma to convert the analytic solution in another format to avoid computing of matrix inversion
- RLS has faster converge speed and suitable for online learning/sequential estimation

Revisit Least Square Estimation: Recursive Least Square for Online/Sequential Estimation

We know the recursive least square algorithm is

RLS algorithm

- $\mathbf{P}_{l+1} = \mathbf{P}_l - \frac{\mathbf{P}_l \mathbf{x}^{(l+1)} {\mathbf{x}^{(l+1)}}^T \mathbf{P}_l}{1 + {\mathbf{x}^{(l+1)}}^T \mathbf{P}_l \mathbf{x}^{(l+1)}}$
- $\widehat{\mathbf{r}}(l+1) = \widehat{\mathbf{r}}(l) + {\mathbf{x}^{(l+1)}}^T y^{(l+1)}$
- $\mathbf{w}(l+1) = \mathbf{P}_{l+1} \widehat{\mathbf{r}}(l+1)$

Revisit Least Square Estimation: Recursive Least Square for Online/Sequential Estimation

RLS algorithm in recursive form of $\mathbf{w}(l)$ is as below

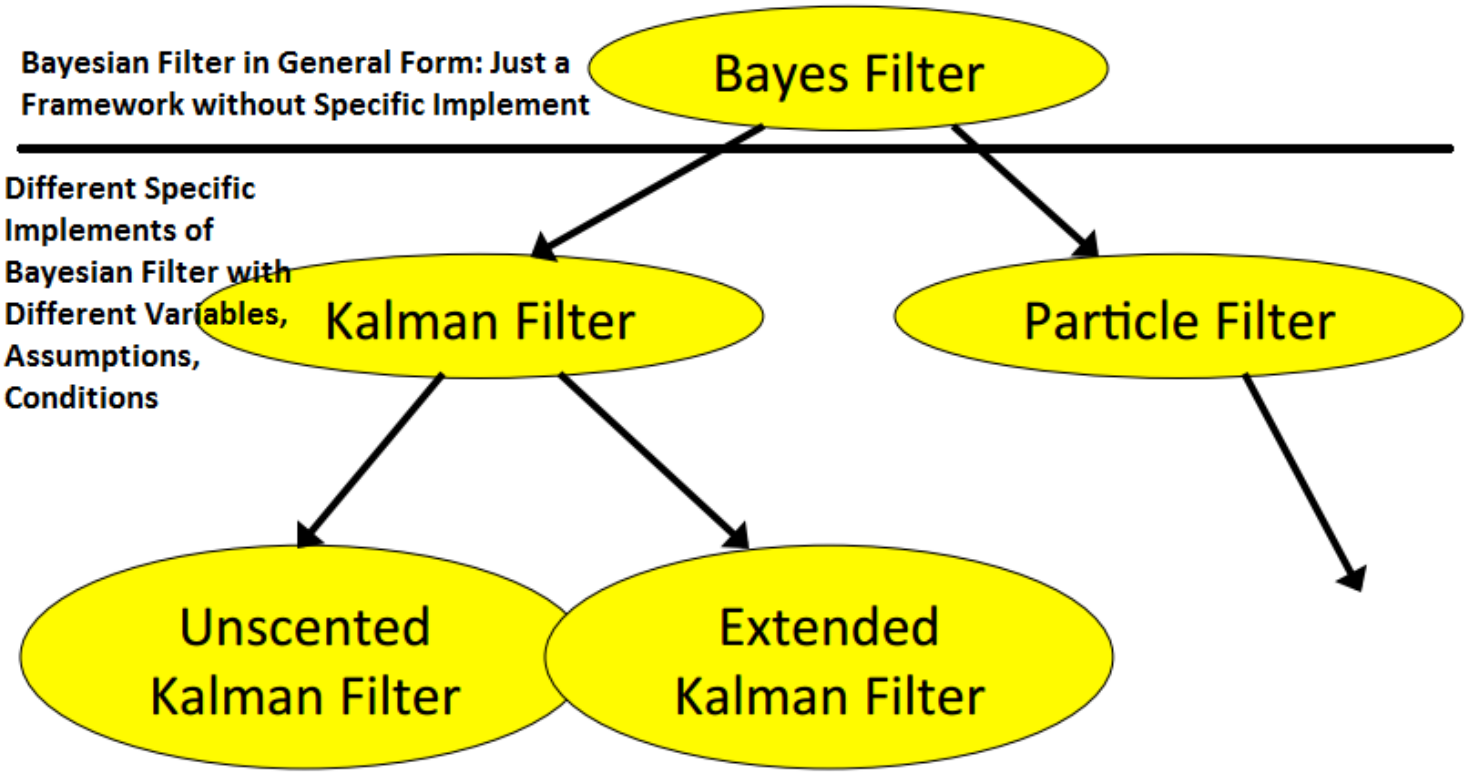
RLS algorithm in recursive form of $\mathbf{w}(l)$

- $\mathbf{k}_{l+1} = \frac{\mathbf{P}_l \mathbf{x}^{(l+1)^T}}{1+\mathbf{x}^{(l+1)} \mathbf{P}_l \mathbf{x}^{(l+1)^T}}$
- $\mathbf{w}(l+1) = \mathbf{w}(l) + \mathbf{k}_{l+1}(y^{(l+1)} - x^{(l+1)}\mathbf{w}(l))$
- $\mathbf{P}_{l+1} = \mathbf{P}_l - \mathbf{k}_{l+1}\mathbf{x}^{(l+1)}\mathbf{P}_l$

For details of derivation of RLS in recursive form of $\mathbf{w}(l)$, please either refer to Lecture 10, Adaptive Signal Processing Course of Tampere University of Technology by Ioan Tabus, wiki Recursive Least Square Filter, or section 4 of Python Machine Learning: The 1st Book Circle

From Gaussian Static to Gaussian Dynamic State System: RLS vs Kalman

We still use figure from Ashutosh Saxena's course Robot Learning Course in Cornell University.



RLS is Part of Kalman Filter

Let us have a quick look of **Kalman Filter**(Here I used the same equations from table 1.1 summary of kalman filter from page 10 in Haykin's book 'Kalman Filtering and Neural Network') without explaining too much regarding what they are, but we come back to analyze details later

- Firstly we define the state space model

State Space Model

- $\mathbf{x}_k = \mathbf{F}_{k,k-1}\mathbf{x}_{k-1} + \mathbf{n}_{k-1}$ (This is called "motion model")
- $\mathbf{y}_k = \mathbf{H}_k\mathbf{x}_k + \mathbf{v}_k$ (This is called "observation model")

- Secondly we have propagation/predication and measurement update equations

From Static to Dynamic State System: RLS to Kalman

Propagation/Predication

- State estimation propagation/predication
  $\widehat{\mathbf{x}}_k^- = \mathbf{F}_{k,k-1}\widehat{\mathbf{x}}_{k-1}^-$
- Error covariance propagation
  $\mathbf{P}_k^- = \mathbf{F}_{k,k-1}\mathbf{P}_{k-1}\mathbf{F}_{k,k-1}^T + \mathbf{Q}_{k-1}$

Measurement Update

- Kalman gain matrix
  $\mathbf{G}_k = \mathbf{P}_k^-\mathbf{H}_k^T(\mathbf{H}_k\mathbf{P}_k^-\mathbf{H}_k^T + \mathbf{R}_k)^{-1}$
- State estimation update
  $\widehat{\mathbf{x}}_k = \widehat{\mathbf{x}}_k^- + \mathbf{G}_k(\mathbf{y}_k - \mathbf{H}_k\widehat{\mathbf{x}}_k^-)$
- Error covariance update
  $\mathbf{P}_k = \mathbf{P}_k^- - \mathbf{G}_k\mathbf{H}_k\mathbf{P}_k^-$

From Recursive Least Square Algorithm to Kalman

Now let us compare 'measurement update' with RLS algorithm

RLS algorithm in recursive form of $\mathbf{w}(l)$

- $\mathbf{k}_{l+1} = \frac{\mathbf{P}_l \mathbf{x}^{(l+1)^T}}{1+\mathbf{x}^{(l+1)}\mathbf{P}_l \mathbf{x}^{(l+1)^T}}$
- $\mathbf{w}(l+1) = \mathbf{w}(l) + \mathbf{k}_{l+1}(y^{(l+1)} - x^{(l+1)}\mathbf{w}(l))$
- $\mathbf{P}_{l+1} = \mathbf{P}_l - \mathbf{k}_{l+1}\mathbf{x}^{(l+1)}\mathbf{P}_l$

Measurement Update

- Kalman gain matrix
  $\mathbf{G}_k = \mathbf{P}_k^- \mathbf{H}_k^T (\mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^T + \mathbf{R}_k)^{-1}$
- State estimation update
  $\widehat{\mathbf{x}}_k = \widehat{\mathbf{x}}_k^- + \mathbf{G}_k(\mathbf{y}_k - \mathbf{H}_k \widehat{\mathbf{x}}_k^-)$
- Error covariance update
  $\mathbf{P}_k = \mathbf{P}_k^- - \mathbf{G}_k \mathbf{H}_k \mathbf{P}_k^-$

From Recursive Least Square Algorithm to Kalman

They are same. The measurement update process of Kalman filter is indeed a RLS adaptive filter. It means RLS is Part of Kalman Filter. Why?

- Now revisit second equation of state space model of Kalman filter

Measument Equation(Observation Model) of State Space Model

$$\mathbf{y}_k = \mathbf{H}_k\mathbf{x}_k + \mathbf{v}_k$$

- The second equation there, the measurement model is actual happened in a linear Gaussian static system(in the measurement, x does NOT change) which is actual a linear regression model in our machine learning problem.

From Recursive Least Square Algorithm to Kalman

- On the other hand, if we look at from the Bayesian filter point of view, the "update step/part" contains the unknown part "likelihood measurement $P(\mathbf{z}_t|\mathbf{x}_t)$" to be decided(as we already got prior distribution $P(\mathbf{x}_t|\mathbf{z}_{1:t-1}) = \int P(\mathbf{x}_t|\mathbf{x}_{t-1})P(\mathbf{x}_{t-1}|\mathbf{z}_{1:t-1})\, d\mathbf{x}_{t-1}$ the from "prediction step"). As we discussed in previous pages(and recall we learned in the machine learning 1st book circle), which criteria the linear static Gaussian system(which is a linear regression model)? Yes, maximize likelihood! And what could be used to maximize likelihood when the observation/measurements are NOT all available once(the training data samples are given in a sequence, from time to time)? Yes, RLS!

That is why We actually use RLS estimation in the Kalman filter

Key Points to Understand Kalman Filter

Let's put all things mentioned previously together.

- In Bayesian filter section, we got

Recursive Expression of Bayesian Filter in General Format

- $$P(\mathbf{x}_t|\mathbf{y}_{1:t}) = \frac{P(\mathbf{y}_{1:t-1})}{P(\mathbf{z}_{1:t})} P(\mathbf{y}_t|\mathbf{x}_t) P(\mathbf{x}_t|\mathbf{y}_{1:t-1}) = \frac{P(\mathbf{y}_{1:t-1})}{P(\mathbf{y}_{1:t})} \cdot P(\mathbf{y}_t|\mathbf{x}_t) \cdot \int P(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{y}_{1:t-1}) P(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1}) \, d\mathbf{x}_{t-1} = \underbrace{\frac{P(\mathbf{y}_{1:t-1})}{P(\mathbf{y}_{1:t})} \cdot P(\mathbf{y}_t|\mathbf{x}_t) \cdot \underbrace{\int P(\mathbf{x}_t|\mathbf{x}_{t-1}) P(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1}) \, d\mathbf{x}_{t-1}}_{prediction}}_{update}$$

Key Points to Understand Kalman Filter

- Later in Kalman filter section, we had the object state space model

State Space Model

- $\mathbf{x}_k = \mathbf{F}_{k,k-1}\mathbf{x}_{k-1} + \mathbf{n}_{k-1}$ (This is called "motion model")
- $\mathbf{y}_k = \mathbf{H}_k\mathbf{x}_k + \mathbf{v}_k$ (This is called "observation model")

- We also saw the Kalman filter algorithm

Key Points to Understand Kalman Filter

Propagation/Predication

- State estimation propagation/predication
  $\widehat{\mathbf{x}}_k^- = \mathbf{F}_{k,k-1}\widehat{\mathbf{x}}_{k-1}^-$
- Error covariance propagation
  $\mathbf{P}_k^- = \mathbf{F}_{k,k-1}\mathbf{P}_{k-1}\mathbf{F}_{k,k-1}^T + \mathbf{Q}_{k-1}$

Measurement Update

- Kalman gain matrix
  $\mathbf{G}_k = \mathbf{P}_k^-\mathbf{H}_k^T(\mathbf{H}_k\mathbf{P}_k^-\mathbf{H}_k^T + \mathbf{R}_k)^{-1}$
- State estimation update
  $\widehat{\mathbf{x}}_k = \widehat{\mathbf{x}}_k^- + \mathbf{G}_k(\mathbf{y}_k - \mathbf{H}_k\widehat{\mathbf{x}}_k^-)$
- Error covariance update
  $\mathbf{P}_k = \mathbf{P}_k^- - \mathbf{G}_k\mathbf{H}_k\mathbf{P}_k^-$

Key Points to Understand Kalman Filter

- The "predication" in Kalman filter algorithm is the solution for solving the estimation problem of "motion model" in state space equations.
- The "measurement update" in Kalman filter algorithm is the solution for solving the estimation problem of "observation model" in state space equations.
- The "predication" in Kalman filter algorithm represents the prior information in Bayes rule
- The core part of "measurement update" in Kalman filter is solving the estimation problem of "observation model", which represents the likelihood in Bayes rule
- So the combination(multiply) of "predication" and "measurement update" is actually calculating the posterior in Bayes rule

Conclusion of Kalman Filter

- Kalman filter could be seen as an extension of "RLS algorithm works for estimation problem of a static linear Gaussian system with maximize likelihood criteria", it uses maximize posterior criteria for estimation problem of a dynamic linear Gaussian system recursively
- Kalman filter works for estimation of random state $\mathbf{x}$ in linear dynamic system which follows Gaussian distribution

Other Bayesian Filters: Extended Kalmen Filter, Unscented Kalmen Filter and Particle Filter

- From here starting to introduce other relative Bayesian filters(Extended Kalmen Filter, Unscented Kalmen Filter and Particle Filter) when the dynamic state model is NOT linear and/or NOT Gaussian
- If you want to know more about Bayesian filter, EKF, UKF, particle filter, etc., I recommend reading Simo Sarkka's book Bayesian Filtering and Smoothing and his course Nonlinear Filtering and Estimation, for tutorial of EKF UKF and particle filter Zhe Chen's paper "Bayesian Filtering: From Kalman Filters to Particle Filters, and Beyond" is a good resource
- We will try to go through every algorithm, also show the mathematics behind each algorithm.

From Linear to Gaussian Driven Nonlinear Dynamic State System

- Now let's consider another system which is little bit different, comparing with linear Gaussian dynamic state system. In the new system, so called Gaussian driven nonlinear dynamic state system, both state transition and observation become nonlinear function as following:

Gaussian Driven Nonlinear (Markov) Dynamic State System Model

- $\mathbf{x}_t = \mathbf{f}(\mathbf{x}_{t-1}) + \mathbf{q}_{t-1}$
- $\mathbf{y}_t = \mathbf{h}(\mathbf{x}_t) + \mathbf{r}_t$
- $\mathbf{q}_{t-1} \sim \mathcal{N}(0, \mathbf{Q}_{t-1})$ is process noise
- $\mathbf{r}_t \sim \mathcal{N}(0, \mathbf{R}_t)$ is measurement noise
- $\mathbf{f}()$ is dynamic model function and $\mathbf{h}()$ measurement function

From Linear to Nonlinear Gaussian Dynamic State System

- Clearly, Kalman filter will NOT work for Gaussian driven nonlinear dynamic state system
- Naturally we ask question, how to deal with nonlinear dynamic model and measurement functions?
- The simple idea is try to convert the nonlinear functions into linear functions, how?
- Several ways, e.g. linear piece wise functions to replace the nonlinear function, Taylor expansion to approximate the nonlinear function, etc.
- For Extended Kalman Filter(EKF), we actually achieve linearization of nonlinear function by using Taylor expansion to approximate the nonlinear function

EKF: Linearization of Nonlinear Dynamic State System in Gaussian Distribution by 1st Order Taylor Expansion

As we discussed on the previous page, EKF linearizes the nonlinear dynamic state system in Gaussian distribution by using 1st order Taylor expansion. From this sub section, we will see how the EKF is derivate and how EKF works

- Firstly, let's recall how to use Taylor expansion to approximate an arbitrary function g(x)

Taylor Theorem in One Real Variable

- Let $k \geq 1$ be an integer, if the function $g(x)$ can be k times differentiable at the point a
- we have $g(x) = g(a) + g'(a)(x-a) + \frac{g''(a)}{2!}(x-a)^2 + ... + \frac{g^{(k)}(a)}{k!}(x-a)^k + h_k(x)(x-a)^k$
  where $\lim_{x \to a} h_k(x) = 0$

EKF: Linearization of Nonlinear Dynamic State System in Gaussian Distribution by 1st Order Taylor Expansion

- If we only use the first and second parts of Taylor expansion to approximate $g(x)$, which has a formal name 1st Order Taylor Expansion. Then obviously we will have $g(x) \approx g(a) + g'(a)(x-a)$ that is using a linear function $g(a) + g'(a)(x-a)$ to approximate the original nonlinear function $g(x)$. This is how we do linearization by using 1st order of Taylor expansion
- By doing this linearization for both dynamic model function $\mathbf{f}()$ and measurement function $\mathbf{h}()$, we could derivate something similar to Kalman filter from Bayesian filter, which is called Extended Kalman filter

EKF: Linearization of Nonlinear Dynamic State System in Gaussian Distribution by 1st Order Taylor Expansion

- By using 1st Order Taylor Expansion on both dynamic model function $\mathbf{f}()$ and measurement function $\mathbf{h}()$, we will get an approximation of both function:

1st Order Taylor Expansion Approximation of Dynamic Model Function $\mathbf{f}()$ and Measurement Function $\mathbf{h}()$

- $\mathbf{f}(\mathbf{x}) \approx \mathbf{f}(\mathbf{m}) + \mathbf{F}_{\mathbf{x}}(\mathbf{m})(\mathbf{x} - \mathbf{m})$
- $\mathbf{h}(\mathbf{x}) \approx \mathbf{h}(\mathbf{m}) + \mathbf{H}_{\mathbf{x}}(\mathbf{m})(\mathbf{x} - \mathbf{m})$
- where $\mathbf{x} \sim \mathcal{N}(\mathbf{m}, \mathbf{P})$
- $\mathbf{F}_{\mathbf{x}}()$ is Jacobian matrix of $\mathbf{f}(\mathbf{x})$, stands for a matrix consists of 1st order partial derivatives of function $\mathbf{f}(\mathbf{x})$
- $\mathbf{H}_{\mathbf{x}}()$ is Jacobian matrix of $\mathbf{h}(\mathbf{x})$

EKF: Linearization of Nonlinear Dynamic State System in Gaussian Distribution by 1st Order Taylor Expansion

- Note only the first item $\mathbf{f}(\mathbf{m})$ and $\mathbf{h}(\mathbf{m})$ contribute to the approximation mean value of function $\mathbf{f}(\mathbf{x})$, $\mathbf{h}(\mathbf{x})$.
  So we actually have $\mathbb{E}[\mathbf{f}(\mathbf{x})] \approx \mathbb{E}[\mathbf{f}(\mathbf{m}) + \mathbf{F}_{\mathbf{x}}(\mathbf{m})(\mathbf{x} - \mathbf{m})] = \mathbb{E}[\mathbf{f}(\mathbf{m})] = \mathbf{f}(\mathbf{m})$ due that $\mathbb{E}[\mathbf{x}] = 0$, $\mathbb{E}[\mathbf{h}(\mathbf{x})] \approx \mathbf{h}(\mathbf{m})$ for same reason
- The second item $\mathbf{F}_{\mathbf{x}}(\mathbf{m})(\mathbf{x} - \mathbf{m})$ and $\mathbf{H}_{\mathbf{x}}(\mathbf{m})(\mathbf{x} - \mathbf{m})$ have 0 mean, they decide the approximation covariance value of function $\mathbf{f}(\mathbf{x})$, $\mathbf{h}(\mathbf{x})$.
  So we actually have
  $Cov[\mathbf{f}(\mathbf{x})] = \mathbb{E}[(\mathbf{f}(\mathbf{x}) - \mathbb{E}[\mathbf{f}(\mathbf{x})])(\mathbf{f}(\mathbf{x}) - \mathbb{E}[\mathbf{f}(\mathbf{x})])^T] \approx \mathbb{E}[(\mathbf{f}(\mathbf{x}) - \mathbf{f}(\mathbf{m}))(\mathbf{f}(\mathbf{x}) - \mathbf{f}(\mathbf{m}))^T] = \mathbf{F}_{\mathbf{x}}(\mathbf{m})\mathbf{P}\mathbf{F}_{\mathbf{x}}^T(\mathbf{m})$, and
  $Cov[\mathbf{f}(\mathbf{x})] \approx \mathbf{H}_{\mathbf{x}}(\mathbf{m})\mathbf{P}\mathbf{H}_{\mathbf{x}}^T(\mathbf{m})$ for same reason

EKF: Linearization of Nonlinear Dynamic State System in Gaussian Distribution by 1st Order Taylor Expansion

- So the approximation of Gaussian driven nonlinear dynamic state system as following:

Linearized Gaussian Driven Nonlinear Dynamic State System Model

- $\mathbf{x}_t = \mathbf{f}(\mathbf{m}) + \mathbf{F}_{\mathbf{x}}(\mathbf{m})(\mathbf{x}_{t-1} - \mathbf{m}) + \mathbf{q}_{t-1}$
- $\mathbf{y}_t = \mathbf{h}(\mathbf{m}) + \mathbf{H}_{\mathbf{x}}(\mathbf{m})(\mathbf{x}_t - \mathbf{m}) + \mathbf{r}_t$
- $\mathbf{q}_{t-1} \sim \mathcal{N}(0, \mathbf{Q}_{t-1})$ is process noise
- $\mathbf{r}_t \sim \mathcal{N}(0, \mathbf{R}_t)$ is measurement noise
- $\mathbf{F}_{\mathbf{x}}()$ and $\mathbf{H}_{\mathbf{x}}()$ are Jacobian matrix of $\mathbf{f}(\mathbf{x})$, $\mathbf{h}(\mathbf{x})$

Note: **In fact, nonlinear transform from Gaussian distribution will NOT be a Gaussian**. So we just assume $\mathbf{x}_t$ and $\mathbf{y}_t$ are Gaussian cause using linearization to replace of nonlinear functions.

EKF: Linearization of Nonlinear Dynamic State System in Gaussian Distribution by 1st Order Taylor Expansion

Same as we derivate for Kalman filter, we assume

- $P(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1}) = \mathcal{N}(\mathbf{m}_{t-1}, \mathbf{P}_{t-1})$ which denotes the probability of state $\mathbf{x}_t$ based on observations from 1 to t
- $P(\mathbf{x}_t|\mathbf{y}_{1:t-1}) = \mathcal{N}(\mathbf{m}_t^-, \mathbf{P}_t^-)$ which denotes the probability of state $\mathbf{x}_t$ based on observations from 1 to t-1
- We do the same thing as for Kalman filter, we could apply lemma 1 on $\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1}$ and $\mathbf{x}_t|\mathbf{y}_{1:t-1}$ by define $\mathbf{a} = \mathbf{x}_{t-1}|\mathbf{y}_{1:t-1}$ and $\mathbf{b} = \mathbf{x}_t|\mathbf{y}_{1:t-1}$
- Then we have $\mathbf{b}|\mathbf{a} = \mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{y}_{1:t-1}$
- Now we know $P(\mathbf{a}) = P(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1}) = \mathcal{N}(\mathbf{m}_{t-1}, \mathbf{P}_{t-1})$ and $P(\mathbf{b}|\mathbf{a}) = P(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{y}_{1:t-1}) = P(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t|\mathbf{f}(\mathbf{m_{k-1}}), \mathbf{Q}_{t-1})$

EKF: Linearization of Nonlinear Dynamic State System in Gaussian Distribution by 1st Order Taylor Expansion

- By using rule which is similar to lemma 1 we get joint probability:
  $P(\mathbf{a} \bigcap \mathbf{b}) = P((\mathbf{x}_t|\mathbf{y}_{1:t-1}) \bigcap P(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1})) =$
  $P((\mathbf{x}_t, \mathbf{x}_{t-1}|\mathbf{y}_{1:t-1}))=$

$$\mathcal{N}\left(\begin{pmatrix} \mathbf{m}_{t-1} \\ \mathbf{f}(\mathbf{m}_{t-1}) \end{pmatrix}, \begin{pmatrix} \mathbf{P}_{t-1} & \mathbf{P}_{t-1}\mathbf{F}_{\mathbf{x}}^{T}(\mathbf{m}_{t-1}) \\ \mathbf{F}_{\mathbf{x}}(\mathbf{m}_{t-1})\mathbf{P}_{t-1} & \mathbf{F}_{\mathbf{x}}(\mathbf{m}_{t-1})\mathbf{P}\mathbf{F}_{\mathbf{x}}^{T}(\mathbf{m}_{t-1}) + \mathbf{Q}_{t-1} \end{pmatrix}\right)$$

- also marginal distribution of $\mathbf{b} = \mathbf{x}_t|\mathbf{y}_{1:t-1}$:
  $P(\mathbf{b}) = P(\mathbf{x}_t|\mathbf{y}_{1:t-1}) = \mathcal{N}(\mathbf{m}_t^-, \mathbf{P}_t^-) =$
  $\mathcal{N}(\mathbf{f}(\mathbf{m}_{t-1}), \mathbf{F}_{\mathbf{x}}(\mathbf{m}_{t-1})\mathbf{P}\mathbf{F}_{\mathbf{x}}^{T}(\mathbf{m}_{t-1}) + \mathbf{Q}_{t-1})$
- Note: here the rule used is a little bit modification of lemma 1, because the $P(\mathbf{b}|\mathbf{a}) = \mathcal{N}(\mathbf{x}_t|\mathbf{f}(\mathbf{m}_{t-1}), \mathbf{Q}_{t-1})$ NOT in the format $\mathcal{N}(\mathbf{b}|\mathbf{H}\mathbf{a}, \mathbf{R})$. So the mean of $P(\mathbf{b})$ is changed and covariance is actually calculated by adding up the covariance of $\mathbf{f}(\mathbf{x}_{t-1})$, $\mathbf{F}_{\mathbf{x}}(\mathbf{m})\mathbf{P}\mathbf{F}_{\mathbf{x}}^{T}(\mathbf{m})$, and the covariance of $\mathbf{q}_{t-1}$, $\mathbf{Q}_{t-1}$

EKF: Linearization of Nonlinear Dynamic State System in Gaussian Distribution by 1st Order Taylor Expansion

- Until now, we already got
  $\mathbf{m}_t^- = \mathbf{f}(\mathbf{m}_{t-1})$
  $\mathbf{P}_t^- = \mathbf{F}_\mathbf{x}(\mathbf{m}_{t-1})\mathbf{P}\mathbf{F}_\mathbf{x}^T(\mathbf{m}_{t-1}) + \mathbf{Q}_{t-1}$
  from prediction step by using the similar way of Kalman filter's derivation
- This is prediction step of EKF

Prediction step of Extended Kalman Filter

- $\mathbf{m}_t^- = \mathbf{f}(\mathbf{m}_{t-1})$
- $\mathbf{P}_t^- = \mathbf{F}_\mathbf{x}(\mathbf{m}_{t-1})\mathbf{P}\mathbf{F}_\mathbf{x}^T(\mathbf{m}_{t-1}) + \mathbf{Q}_{t-1}$

EKF: Linearization of Nonlinear Dynamic State System in Gaussian Distribution by 1st Order Taylor Expansion

- Keep doing the similar things for Kalman filter's derivation again. Next we define $\mathbf{a} = \mathbf{x}_t|\mathbf{y}_{1:t-1}$ and $\mathbf{b} = \mathbf{y}_t|\mathbf{y}_{1:t-1}$ and use rule similar to lemma 1 again then lemma 2 (**Here we skip the derivation, please follow how we derivate update step of Kalman filter to derivate by urself**)
- We can get the
  $\mathbf{m}_t = \mathbf{m}_t^- + \mathbf{P}_t^- \mathbf{H}_x^T(\mathbf{m}_t^-)(\mathbf{H}_x(\mathbf{m}_t^-)\mathbf{P}_t^- \mathbf{H}_x^T(\mathbf{m}_t^-) + \mathbf{R}_t)^{-1}(\mathbf{y}_t - \mathbf{h}(\mathbf{m}_t^-))$
  and $\mathbf{P}_t = \mathbf{P}_t^- - \mathbf{P}_t^- \mathbf{H}_x^T(\mathbf{m}_t^-)(\mathbf{H}_x(\mathbf{m}_t^-)\mathbf{P}_t^- \mathbf{H}_x^T(\mathbf{m}_t^-) + \mathbf{R}_t)^{-1}\mathbf{H}_x(\mathbf{m}_t^-)\mathbf{P}_t^-$
  which came from update step

EKF: Linearization of Nonlinear Dynamic State System in Gaussian Distribution by 1st Order Taylor Expansion

- This is update step of EKF

Measurements Update Step of Extended Kalman Filter

- $\mathbf{S}_t = \mathbf{H}_x(\mathbf{m}_t^-)\mathbf{P}_t^-\mathbf{H}_x^T(\mathbf{m}_t^-) + \mathbf{R}_t$
- $\mathbf{v}_t = \mathbf{y}_t - \mathbf{h}(\mathbf{m}_t^-)$
- $\mathbf{K}_t = \mathbf{P}_t^-\mathbf{H}_x^T(\mathbf{m}_t^-)\mathbf{S}_t^{-1}$
- $\mathbf{m}_t = \mathbf{m}_t^- + \mathbf{K}_t\mathbf{v}_t$
- $\mathbf{P}_t = \mathbf{P}_t^- - \mathbf{K}_t\mathbf{S}_t\mathbf{K}_t^T$

EKF: Linearization of Nonlinear Dynamic State System in Gaussian Distribution by 1st Order Taylor Expansion

Combine prediction and update, we get Extended Kalman filter

Prediction step of Extended Kalman Filter

- $\mathbf{m}_t^- = \mathbf{f}(\mathbf{m}_{t-1})$
- $\mathbf{P}_t^- = \mathbf{F}_\mathbf{x}(\mathbf{m}_{t-1})\mathbf{P}\mathbf{F}_\mathbf{x}^T(\mathbf{m}_{t-1}) + \mathbf{Q}_{t-1}$

Measurements Update Step of Extended Kalman Filter

- $\mathbf{S}_t = \mathbf{H}_x(\mathbf{m}_t^-)\mathbf{P}_t^-\mathbf{H}_x^T(\mathbf{m}_t^-) + \mathbf{R}_t$
- $\mathbf{v}_t = \mathbf{y}_t - \mathbf{h}(\mathbf{m}_t^-)$
- $\mathbf{K}_t = \mathbf{P}_t^-\mathbf{H}_x^T(\mathbf{m}_t^-)\mathbf{S}_t^{-1}$
- $\mathbf{m}_t = \mathbf{m}_t^- + \mathbf{K}_t\mathbf{v}_t$
- $\mathbf{P}_t = \mathbf{P}_t^- - \mathbf{K}_t\mathbf{S}_t\mathbf{K}_t^T$

EKF Algorithm(Start from t = 1)

for t in T **do**
    // Prediction step of Extended Kalman Filter
    $\mathbf{x}_t^- = \mathbf{f}(\mathbf{x}_{t-1})$
    $\mathbf{P}_t^- = \mathbf{F}_{\mathbf{x}}(\mathbf{x}_{t-1})\mathbf{P}\mathbf{F}_{\mathbf{x}}^T(\mathbf{x}_{t-1}) + \mathbf{Q}_{t-1}$
    // Measurements Update Step of Extended Kalman Filter
    $\mathbf{S}_t = \mathbf{H}_x(\mathbf{x}_t^-)\mathbf{P}_t^-\mathbf{H}_x^T(\mathbf{x}_t^-) + \mathbf{R}_t$
    $\mathbf{v}_t = \mathbf{y}_t - \mathbf{h}(\mathbf{x}_t^-)$
    $\mathbf{K}_t = \mathbf{P}_t^-\mathbf{H}_x^T(\mathbf{x}_t^-)\mathbf{S}_t^{-1}$
    $\mathbf{x}_t = \mathbf{x}_t^- + \mathbf{K}_t\mathbf{v}_t$
    $\mathbf{P}_t = \mathbf{P}_t^- - \mathbf{K}_t\mathbf{S}_t\mathbf{K}_t^T$
    t = t + 1
end for

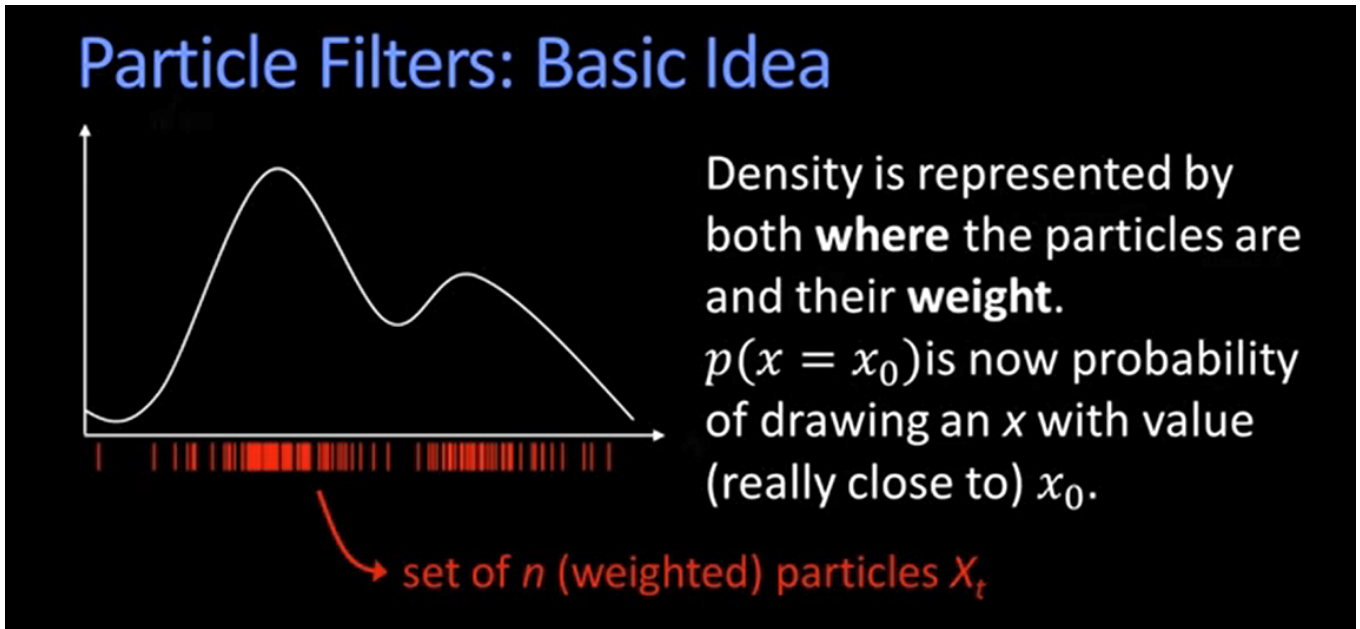
Importance Sampling: Way to Approximate Arbitrary Distribution

- Now let's consider the case where dynamic state system is Non-linear and Non-Gaussian, which means the distribution of random variable $\mathbf{x}$ can be anything. We define this "arbitrary" distribution of random variable $\mathbf{x}$ as $p(\mathbf{x})$
- According to the definition of Bayesian filter in general, $P(\mathbf{x}_t|\mathbf{y}_{1:t}) = \frac{P(\mathbf{y}_{1:t-1})}{P(\mathbf{z}_{1:t})} P(\mathbf{y}_t|\mathbf{x}_t) P(\mathbf{x}_t|\mathbf{y}_{1:t-1})$, suppose we would like to know $\mathbb{E}[f(\mathbf{x}_t)] = \int f(\mathbf{x}_t) p(\mathbf{x}_t|\mathbf{y}_{1:t})\, dx$, where note $p(\mathbf{x}_t|\mathbf{y}_{1:t})$ is "arbitrary" unknown distribution(NOT Gaussian or any known, easy to represent distribution)
- So the problem is, we do NOT know $p(\mathbf{x}_t|\mathbf{y}_{1:t})$ and it is NOT easy to represent this "arbitrary" unknown distribution due it is NOT Gaussian anymore

Importance Sampling: Way to Approximate Arbitrary Distribution

How to represent this "arbitrary" unknown distribution $p(\mathbf{x}_t|\mathbf{y}_{1:t})$?

- Answer is, by using particles in Monte Carlo(MC) framework
- Although we do NOT know the "shape" of this arbitrary unknown distribution, but in case we could have very large amount of samples which follows this arbitrary unknown distribution, what we could get? Let's see the figure



Importance Sampling: Way to Approximate Arbitrary Distribution

How to represent this "arbitrary" unknown distribution $p(\mathbf{x}_t|\mathbf{y}_{1:t})$?

- Obviously, the density of samples/particles reflects the "shape" of distribution function, the more particles in a certain value the higher output of distribution function $p(\mathbf{x}_t|\mathbf{y}_{1:t})$ in that certain value of $\mathbf{x}_t^{(i)}$, where i stands for i^{th} particle
- In the other words, the total number of samples/particles shown on the certain value $\mathbf{x}_t^{(i)}$(Note: It for sure could be more than 1 particles shown on the same certain value $\mathbf{x}_t^{(i)}$, which means $\mathbf{x}_t^{(i)}$ can be same as $\mathbf{x}_t^{(j)}$ where $i \neq j$) divides the total number of particles, N, equals to $p(\mathbf{x}_t|\mathbf{y}_{1:t})$ approximately, in case N is large enough

Importance Sampling: Way to Approximate Arbitrary Distribution

- That is
  $p(\mathbf{x}_t|\mathbf{y}_{1:t}) \approx \hat{p}(\mathbf{x}_t|\mathbf{y}_{1:t}) = \frac{1}{N}\sum_{i=1}^{N}\delta(\mathbf{x}_t - \mathbf{x}_t^{(i)})$
- where $\delta(\mathbf{x}_t - \mathbf{x}_t^{(i)})$ is Dirac function, means if $\mathbf{x}_t = \mathbf{x}_t^{(i)}$ the output is 1, otherwise output is 0.
- It is easy to understand this approximation of $p(\mathbf{x}_t|\mathbf{y}_{1:t})$, is done by counting the portion of total number of particles which dropped on the same position, $\mathbf{x}_t^{(i)}$(as we discussed, more than one particles can have the same value), out of the total number of particles, N. This portion is approximately equivalent to $p(\mathbf{x}_t|\mathbf{y}_{1:t})$ if N is large

Importance Sampling: Way to Approximate Arbitrary Distribution

So we go back to our original problem, what is $\mathbb{E}[f(\mathbf{x}_t)]$?

- By using $p(\mathbf{x}_t|\mathbf{y}_{1:t}) \approx \hat{p}(\mathbf{x}_t|\mathbf{y}_{1:t}) = \frac{1}{N}\sum_{i=1}^{N}\delta(\mathbf{x}_t - \mathbf{x}_t^{(i)})$, we can get
- $\mathbb{E}[f(\mathbf{x}_t)] = \int f(\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{y}_{1:t})\,d\mathbf{x}_t \approx \int f(\mathbf{x}_t)\hat{p}(\mathbf{x}_t|\mathbf{y}_{1:t})\,d\mathbf{x}_t = \frac{1}{N}\sum_{i=1}^{N}\int f(\mathbf{x}_t)\delta(\mathbf{x}_t - \mathbf{x}_t^{(i)})\,d\mathbf{x}_t = \frac{1}{N}\sum_{i=1}^{N} f(\mathbf{x}_t^{(i)})$
- It means we get

$$\mathbb{E}[f(\mathbf{x}_t)] = \int f(\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{y}_{1:t})\,d\mathbf{x}_t \approx \frac{1}{N}\sum_{i=1}^{N} f(\mathbf{x}_t^{(i)})$$

Importance Sampling: Way to Approximate Arbitrary Distribution

However we noticed the particles should be sampled from/by following this "arbitrary" unknown distribution $p(\mathbf{x}_t|\mathbf{y}_{1:t})$, but the problem is, we do NOT know this "arbitrary" unknown distribution $p(\mathbf{x}_t|\mathbf{y}_{1:t})$, how to sample particles which follows this distribution?

- In order to find some way to calculate $\mathbb{E}[f(\mathbf{x}_t)]$, we introduce "importance sampling" as follow:

Importance Sampling:

- Suppose we have a known distribution(e.g. Gaussian) $q(\mathbf{x}_t|\mathbf{y}_{1:t})$
- Let's see how to use this known distribution to calculate $\mathbb{E}[f(\mathbf{x}_t)]$

Importance Sampling: Way to Approximate Arbitrary Distribution

- $\mathbb{E}[f(\mathbf{x}_t)] = \int f(\mathbf{x}_t) p(\mathbf{x}_t|\mathbf{y}_{1:t})\, d\mathbf{x}_t =$

  $\int f(\mathbf{x}_t) p(\mathbf{x}_t|\mathbf{y}_{1:t}) \cdot q(\mathbf{x}_t|\mathbf{y}_{1:t}) / q(\mathbf{x}_t|\mathbf{y}_{1:t})\, d\mathbf{x}_t =$

  $\int f(\mathbf{x}_t) \frac{p(\mathbf{y}_{1:t}|\mathbf{x}_t) p(\mathbf{x}_t)}{p(\mathbf{y}_{1:t})} \cdot q(\mathbf{x}_t|\mathbf{y}_{1:t}) / q(\mathbf{x}_t|\mathbf{y}_{1:t})\, d\mathbf{x}_t =$

  $\frac{1}{p(\mathbf{y}_{1:t})} \int f(\mathbf{x}_t) \frac{p(\mathbf{y}_{1:t}|\mathbf{x}_t) p(\mathbf{x}_t)}{q(\mathbf{x}_t|\mathbf{y}_{1:t})} \cdot q(\mathbf{x}_t|\mathbf{y}_{1:t})\, d\mathbf{x}_t,$

  due to Bayes rule $p(\mathbf{x}_t|\mathbf{y}_{1:t}) p(\mathbf{y}_{1:t}) = p(\mathbf{y}_{1:t}|\mathbf{x}_t) p(\mathbf{x}_t)$
- Now define $W_t(\mathbf{x}_t) = \frac{p(\mathbf{y}_{1:t}|\mathbf{x}_t) p(\mathbf{x}_t)}{q(\mathbf{x}_t|\mathbf{y}_{1:t})}$
- So we get $\mathbb{E}[f(\mathbf{x}_t)] = \frac{1}{p(\mathbf{y}_{1:t})} \int f(\mathbf{x}_t) W_t(\mathbf{x}_t) \cdot q(\mathbf{x}_t|\mathbf{y}_{1:t})\, d\mathbf{x}_t$

Importance Sampling: Way to Approximate Arbitrary Distribution

- Notice $p(\mathbf{y}_{1:t}) = \int p(\mathbf{x}_t \cap \mathbf{y}_{1:t})\, d\mathbf{x}_t = \int p(\mathbf{y}_{1:t}|\mathbf{x}_t)p(\mathbf{x}_t)\, d\mathbf{x}_t$
- So origin could be rewritten as:

$$\mathbb{E}[f(\mathbf{x}_t)] = \frac{1}{p(\mathbf{y}_{1:t})} \int f(\mathbf{x}_t) W_t(\mathbf{x}_t) \cdot q(\mathbf{x}_t|\mathbf{y}_{1:t})\, d\mathbf{x}_t =$$

$$\frac{\int f(\mathbf{x}_t) W_t(\mathbf{x}_t) \cdot q(\mathbf{x}_t|\mathbf{y}_{1:t})\, d\mathbf{x}_t}{\int p(\mathbf{y}_{1:t}|\mathbf{x}_t)p(\mathbf{x}_t)\, d\mathbf{x}_t} =$$

$$\frac{\int f(\mathbf{x}_t) W_t(\mathbf{x}_t) \cdot q(\mathbf{x}_t|\mathbf{y}_{1:t})\, d\mathbf{x}_t}{\int \frac{p(\mathbf{y}_{1:t}|\mathbf{x}_t)p(\mathbf{x}_t)}{q(\mathbf{x}_t|\mathbf{y}_{1:t})} \cdot q(\mathbf{x}_t|\mathbf{y}_{1:t})\, d\mathbf{x}_t}$$

Importance Sampling: Way to Approximate Arbitrary Distribution

- Cause we have $W_t(\mathbf{x}_t) = \frac{p(\mathbf{y}_{1:t}|\mathbf{x}_t)p(\mathbf{x}_t)}{q(\mathbf{x}_t|\mathbf{y}_{1:t})}$
- It means $\mathbb{E}[f(\mathbf{x}_t)] = \frac{\int f(\mathbf{x}_t) W_t(\mathbf{x}_t) \cdot q(\mathbf{x}_t|\mathbf{y}_{1:t})\, d\mathbf{x}_t}{\int \frac{p(\mathbf{y}_{1:t}|\mathbf{x}_t)p(\mathbf{x}_t)}{q(\mathbf{x}_t|\mathbf{y}_{1:t})} \cdot q(\mathbf{x}_t|\mathbf{y}_{1:t})\, d\mathbf{x}_t} =$

$$\frac{\int f(\mathbf{x}_t) W_t(\mathbf{x}_t) \cdot q(\mathbf{x}_t|\mathbf{y}_{1:t})\, d\mathbf{x}_t}{\int W_t(\mathbf{x}_t) \cdot q(\mathbf{x}_t|\mathbf{y}_{1:t})\, d\mathbf{x}_t}$$

- Note $q(\mathbf{x}_t|\mathbf{y}_{1:t})$ is the distribution we know, that means we could sample particle from this known distribution easily.

Importance Sampling: Way to Approximate Arbitrary Distribution

- Recall the conclusion we got from previous derivation, $\mathbb{E}[f(\mathbf{x}_t)] = \int f(\mathbf{x}_t) p(\mathbf{x}_t|\mathbf{y}_{1:t})\, d\mathbf{x}_t \approx \int f(\mathbf{x}_t) \hat{p}(\mathbf{x}_t|\mathbf{y}_{1:t})\, d\mathbf{x}_t = \frac{1}{N}\sum_{i=1}^{N} \int f(\mathbf{x}_t)\delta(\mathbf{x}_t - \mathbf{x}_t^{(i)})\, d\mathbf{x}_t = \frac{1}{N}\sum_{i=1}^{N} f(\mathbf{x}_t^{(i)})$, where $p(\mathbf{x}_t|\mathbf{y}_{1:t})$ is arbitrary distribution. And now we have a known distribution $q(\mathbf{x}_t|\mathbf{y}_{1:t})$ can be sampled from

Importance Sampling: Way to Approximate Arbitrary Distribution

- Applying conclusion recalled last page, obviously we can get

$$\mathbb{E}[f(\mathbf{x}_t)] = \frac{\int f(\mathbf{x}_t) W_t(\mathbf{x}_t) \cdot q(\mathbf{x}_t|\mathbf{y}_{1:t})\, d\mathbf{x}_t}{\int W_t(\mathbf{x}_t) \cdot q(\mathbf{x}_t|\mathbf{y}_{1:t})\, d\mathbf{x}_t} = \frac{\mathbb{E}[f(\mathbf{x}_t) W_t(\mathbf{x}_t)]}{\mathbb{E}[W_t(\mathbf{x}_t)]} = \frac{\frac{1}{N}\sum_{i=1}^{N} f(\mathbf{x}_t^{(i)}) W_t(\mathbf{x}_t^{(i)})}{\frac{1}{N}\sum_{i=1}^{N} W_t(\mathbf{x}_t^{(i)})}$$

, where $\mathbf{x}_t|\mathbf{y}_{1:t}$ follows known distribution $q(\mathbf{x}_t|\mathbf{y}_{1:t})$

Importance Sampling: Way to Approximate Arbitrary Distribution

- Rewrite the result again

$$\mathbb{E}[f(\mathbf{x}_t)] = \frac{\frac{1}{N}\sum_{i=1}^{N} f(\mathbf{x}_t^{(i)})\, W_t(\mathbf{x}_t^{(i)})}{\frac{1}{N}\sum_{i=1}^{N} W_t(\mathbf{x}_t^{(i)})} = \sum_{i=1}^{N} \widetilde{W}_t(\mathbf{x}_t^{(i)}) f(\mathbf{x}_t^{(i)}),$$

  where $\widetilde{W}_t(\mathbf{x}_t^{(i)}) = \frac{W_t(\mathbf{x}_t^{(i)})}{\sum_{i=1}^{N} W_t(\mathbf{x}_t^{(i)})}$ is called as normalized weight

- Until now, we finished up derivation of "importance sampling", changing the expression from something with an unknown distribution to something with known distribution and represented in form of approximation by particles
- Now the issue we concern on is, how to calculate $W_t(\mathbf{x}_t^{(i)})$? Let's move to next sub section for the answer

Sequential Importance Sampling

From this sub section, we will introduce method to calculate $W_t(\mathbf{x}_t^{(i)})$ in a recursive format, it is **Sequential Importance Sampling**:

- First **assume** our chosen "known distribution" $q(\cdot)$ satisfies: $q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t}) = q(\mathbf{x}_t|\mathbf{x}_{0:t-1}, \mathbf{y}_{1:t})q(\mathbf{x}_{0:t-1}|\mathbf{y}_{1:t-1})$
- Note here we are manipulating with $q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})$ rather than $q(\mathbf{x}_t|\mathbf{y}_{1:t})$

Sequential Importance Sampling

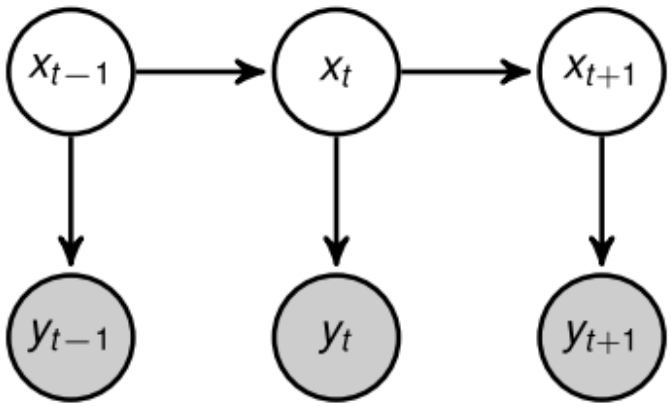
Then let's analyze "arbitrary unknown" distribution $p(\cdot)$

- Similarly, we look at the posterior of all states $\mathbf{x}_{0:t}$ satisfies
  $p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t}) = \frac{p(\mathbf{y}_t|\mathbf{x}_{0:t}, \mathbf{y}_{1:t-1})p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t-1})}{p(\mathbf{y}_t|\mathbf{y}_{1:t-1})}$
  due to Bayes rule $P(B|A) = \frac{P(A|B)P(B)}{P(A)}$ with setting of
  $B \leftarrow \mathbf{x}_{0:t}|\mathbf{y}_{1:t-1}$ and $A \leftarrow \mathbf{y}_t|\mathbf{y}_{1:t-1}$
- Notice $p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t-1}) = p(\mathbf{x}_t|\mathbf{x}_{0:t-1}\mathbf{y}_{1:t-1})p(\mathbf{x}_{0:t-1}|\mathbf{y}_{1:t-1})$ due to Bayes rule $P(B|A)P(A) = P(A \bigcap B)$ with setting of
  $B \leftarrow \mathbf{x}_t|\mathbf{y}_{1:t-1}$ and $A \leftarrow \mathbf{x}_{0:t-1}|\mathbf{y}_{1:t-1}$
- Combine them, we get

$$p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t}) = \frac{p(\mathbf{y}_t|\mathbf{x}_{0:t}, \mathbf{y}_{1:t-1})p(\mathbf{x}_t|\mathbf{x}_{0:t-1}\mathbf{y}_{1:t-1})p(\mathbf{x}_{0:t-1}|\mathbf{y}_{1:t-1})}{p(\mathbf{y}_t|\mathbf{y}_{1:t-1})}$$

Sequential Importance Sampling

- Due to our model is a HMM(Recall HMM in Bayesian filter slide)



- observation $\mathbf{y}_t$ is only depend on hidden state $p(\mathbf{x}_t$(That is "All observations $\mathbf{y}_t$ are conditional independent on state $\mathbf{x}_t$") and hidden state $p(\mathbf{x}_t$ is only dependent on previous state hidden state $p(\mathbf{x}_{t-1}$(That is state $\mathbf{x}_t$ is a Markov process)

Sequential Importance Sampling

- By using these two rules, we have
  $p(\mathbf{y}_t|\mathbf{x}_{0:t}, \mathbf{y}_{1:t-1}) = p(\mathbf{y}_t|\mathbf{x}_t)$ and
  $p(\mathbf{x}_t|\mathbf{x}_{0:t-1}\mathbf{y}_{1:t-1}) = p(\mathbf{x}_t|\mathbf{x}_{t-1})$
- Then we get

$$p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t}) = \frac{p(\mathbf{y}_t|\mathbf{x}_{0:t}, \mathbf{y}_{1:t-1})p(\mathbf{x}_t|\mathbf{x}_{0:t-1}\mathbf{y}_{1:t-1})p(\mathbf{x}_{0:t-1}|\mathbf{y}_{1:t-1})}{p(\mathbf{y}_t|\mathbf{y}_{1:t-1})}$$
$$= \frac{p(\mathbf{y}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{x}_{t-1})p(\mathbf{x}_{0:t-1}|\mathbf{y}_{1:t-1})}{p(\mathbf{y}_t|\mathbf{y}_{1:t-1})}$$
$$\propto p(\mathbf{y}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{x}_{t-1})p(\mathbf{x}_{0:t-1}|\mathbf{y}_{1:t-1})$$

- In the other words
  $p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t}) \propto p(\mathbf{y}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{x}_{t-1})p(\mathbf{x}_{0:t-1}|\mathbf{y}_{1:t-1})$

Sequential Importance Sampling

- Recall we already defined $W_t(\mathbf{x}_t) = \frac{p(\mathbf{y}_{1:t}|\mathbf{x}_t)p(\mathbf{x}_t)}{q(\mathbf{x}_t|\mathbf{y}_{1:t})}$ in the sub section "Importance Sampling"
- So we have $W_t(\mathbf{x}_t) = \frac{p(\mathbf{y}_{1:t}|\mathbf{x}_t)p(\mathbf{x}_t)}{q(\mathbf{x}_t|\mathbf{y}_{1:t})} = \frac{p(\mathbf{x}_t|\mathbf{y}_{1:t})p(\mathbf{y}_{1:t})}{q(\mathbf{x}_t|\mathbf{y}_{1:t})}$
  $\propto \frac{p(\mathbf{x}_t|\mathbf{y}_{1:t})}{q(\mathbf{x}_t|\mathbf{y}_{1:t})}$
- Notice $\frac{p(\mathbf{x}_t|\mathbf{y}_{1:t})}{q(\mathbf{x}_t|\mathbf{y}_{1:t})} \propto \frac{p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})}{q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})}$
- We could get
  $W_t(\mathbf{x}_t) \propto \frac{p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})}{q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})}$ which is $W_t(\mathbf{x}_t^{(i)}) \propto \frac{p(\mathbf{x}_{0:t}^{(i)}|\mathbf{y}_{1:t})}{q(\mathbf{x}_{0:t}^{(i)}|\mathbf{y}_{1:t})}$

Sequential Importance Sampling

- As we already got $p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t}) \propto p(\mathbf{y}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{x}_{t-1})p(\mathbf{x}_{0:t-1}|\mathbf{y}_{1:t-1})$ and assumed $q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t}) = q(\mathbf{x}_t|\mathbf{x}_{0:t-1}, \mathbf{y}_{1:t})q(\mathbf{x}_{0:t-1}|\mathbf{y}_{1:t-1})$
- Then we could derivate as below:

$$W_t(\mathbf{x}_t^{(i)}) \propto \frac{p(\mathbf{x}_{0:t}^{(i)}|\mathbf{y}_{1:t})}{q(\mathbf{x}_{0:t}^{(i)}|\mathbf{y}_{1:t})}$$

$$\propto \frac{p(\mathbf{y}_t|\mathbf{x}_t^{(i)})p(\mathbf{x}_t^{(i)}|\mathbf{x}_{t-1}^{(i)})p(\mathbf{x}_{0:t-1}^{(i)}|\mathbf{y}_{1:t-1})}{q(\mathbf{x}_t^{(i)}|\mathbf{x}_{0:t-1}^{(i)}, \mathbf{y}_{1:t})q(\mathbf{x}_{0:t-1}^{(i)}|\mathbf{y}_{1:t-1})}$$

- Due that $\frac{p(\mathbf{x}_{0:t-1}^{(i)}|\mathbf{y}_{1:t-1})}{q(\mathbf{x}_{0:t-1}^{(i)}|\mathbf{y}_{1:t-1})} = W_{t-1}(\mathbf{x}_{t-1}^{(i)})$
- We could get $W_t(\mathbf{x}_t^{(i)}) = \frac{p(\mathbf{y}_t|\mathbf{x}_t^{(i)})p(\mathbf{x}_t^{(i)}|\mathbf{x}_{t-1}^{(i)})}{q(\mathbf{x}_t^{(i)}|\mathbf{x}_{0:t-1}^{(i)}, \mathbf{y}_{1:t})} W_{t-1}(\mathbf{x}_{t-1}^{(i)})$
- Now we could calculate $W_t(\mathbf{x}_t^{(i)})$ recursively

Sequential Importance Sampling

- And after calculation of $W_t(\mathbf{x}_t^{(i)})$ recursively, do NOT forget to normalize weight by using
  $\widetilde{W}_t(\mathbf{x}_t^{(i)}) = \frac{W_t(\mathbf{x}_t^{(i)})}{\sum_{i=1}^{N} W_t(\mathbf{x}_t^{(i)})}$
- At last step, we could calculate
  $\mathbb{E}[f(\mathbf{x}_t)] = \sum_{i=1}^{N} \widetilde{W}_t(\mathbf{x}_t^{(i)}) f(\mathbf{x}_t^{(i)})$
- All in all, we get particle filter algorithm by sequential importance sampling as in next page:

Sequential Importance Sampling Algorithm(from t=1)

for t in T **do**
  for i in N **do**
    $\mathbf{x}_t^{(i)} \sim q(\mathbf{x}_t^{(i)} | \mathbf{x}_{0:t-1}^{(i)}, \mathbf{y}_{1:t})$ // draw particles follows $q(\cdot)$
    i = i + 1
  end for
  for i in N **do**
    $$W_t(\mathbf{x}_t^{(i)}) = \frac{p(\mathbf{y}_t|\mathbf{x}_t^{(i)}) p(\mathbf{x}_t^{(i)}|\mathbf{x}_{t-1}^{(i)})}{q(\mathbf{x}_t^{(i)}|\mathbf{x}_{0:t-1}^{(i)}, \mathbf{y}_{1:t})} W_{t-1}(\mathbf{x}_{t-1}^{(i)})$$
    // calculate $W_t(\mathbf{x}_t^{(i)})$
    i = i + 1
  end for
  $$\widetilde{W}_t(\mathbf{x}_t^{(i)}) = \frac{W_t(\mathbf{x}_t^{(i)})}{\sum_{i=1}^{N} W_t(\mathbf{x}_t^{(i)})}$$
  $$\mathbb{E}[f(\mathbf{x}_t)] = \sum_{i=1}^{N} \widetilde{W}_t(\mathbf{x}_t^{(i)}) f(\mathbf{x}_t^{(i)})$$
end for

Particle Filter: Monte Carlo Method for Arbitrary Distribution State by Using Sequential Importance Sampling and Resampling

- It seems the sequential importance sampling algorithm works! But, in fact, ...
- The distribution of the importance weights becomes more and more skewed as time increases. After a few iterations of algorithm, only few or one $W_t(\mathbf{x}_t^{(i)})$ will be non-zero. It is often called weight degeneracy
- This is disadvantageous cause a lot of computing effort is wasted to update those trivial weight
- How to solve this problem?
- We will introduce a new concept for measurement of degeneracy, the so-called effective sample size, N_{eff}

Particle Filter: Sequential Importance Sampling and Resampling

- $N_{eff} = \frac{N}{1 + Var_{q(\cdot|\mathbf{y}_{1:t})}[\widetilde{W}_t(\mathbf{x}_{0:t}^{(i)})]} = \frac{N}{\mathbb{E}_{q(\cdot|\mathbf{y}_{1:t})}[\widetilde{W}_t(\mathbf{x}_{0:t}^{(i)})^2]} \leq N$
- where $Var_{q(\cdot|\mathbf{y}_{1:t})}$ stands the variance of random variable which follows the distribution $q(\cdot|\mathbf{y}_{1:t})$
- A smaller N_{eff} means a larger variance for the weights, which means most of particles become very light weight and only few has high weight, hence more degeneracy
- So we usually define a threshold $N_{threshold} = \frac{N}{3}$, or $N_{threshold} = \frac{N}{2}$, as soon as $N_{eff} \leq N_{threshold}$ we do resmapling.
- By doing this resampling together with sequential importance sampling, we get the particle filter algorithm
- In practical we use $\hat{N}_{eff} = \frac{1}{\sum_{i=1}^{N}(\widetilde{W}_t(\mathbf{x}_t^{(i)}))^2}$ to replace N_{eff}

Outline for Section 3

Outline for Section 4

Outline for Section 5

Conclusion

-
-
-
-

Conclusion

References

 Simo Sarkka, Nonlinear Filtering and Estimation Course, Aalto University

 Zhe Chen, Bayesian Filtering: From Kalman Filters to Particle Filters, and Beyond

 Timothy D. Barfoot, State Estimation for Robotics

 Michal Reinštein, From Bayes to Extended Kalman Filter

 Greg Welch, Gary Bishop, An Introduction to the Kalman Filter

 Simon Haykin, Kalman Filtering and Neural Network

 Simo Sarkka, Kalman Bayesian Filtering and Smoothing

References


Dan Simon, Crash Course on Kalman Filtering, Cleveland State University


Wikipedia: Simultaneous Localization and Mapping (SLAM) Link


Ashutosh Saxena, Robot Learning Course, Cornell University


Cyrill Stachniss, Robot Mapping and SLAM Course, University of Freiburg/University of Bonn


Gustaf Hendeby, Sensor Fusion Course, Linköping University


Ioan Tabus, Adaptive Signal Processing Course, Tampere University of Technology


John Paisley, Machine Learning Course, Columbia University/Edx


Paul Michael Newman, EKF Based Navigation and SLAM, Oxford University


Wikipedia: Extended Kalman Filter Link


Wikipedia: Taylor's Theorem Link

Question?