Chapter 03: Filtering

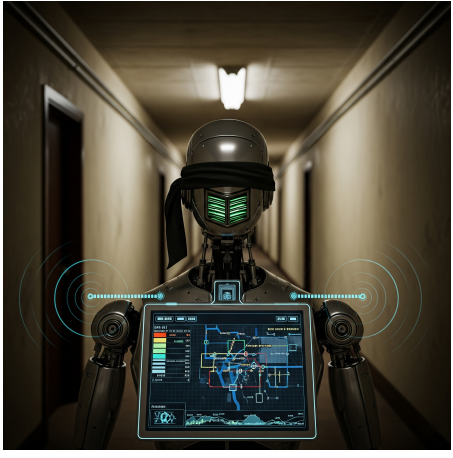
Probabilistic Artificial Intelligence

Notes by Kumar Anurag

The Story of Blindfolded Robot

Imagine:

- A robot is navigating down a hallway, but it's **blindfolded**.
- It can't see where it is — it only has a noisy distance sensor and a map of the hallway.
- At each step, it must decide:
  - Where it *thinks* it is,
  - How to move next,
  - And how to adjust based on new noisy sensor readings.



The robot's challenge is exactly what filtering solves:

"Estimate where you are over time — with incomplete and uncertain information."

What is Filtering? The Core Problem

Filtering is about estimating the **hidden state** of a system over time, using a stream of noisy observations.

Why it's harder than regular regression

- The unknown (state) keeps changing.
- The observations are noisy and only partially informative.
- We must update our estimate *as new data arrives*.

We need:

- A **dynamics model**: How the state evolves over time.
- An **observation model**: How observations relate to the state.

Applications:

- Self-driving cars, speech recognition, finance, robotics, weather prediction...

State Space Models (SSMs)

Let's formalize the blindfolded robot's world.

- **Hidden States:** $X_t \in \mathbb{R}^d$
  - The robot's true position, speed, or internal state at time t.
  - Not directly visible — we want to estimate it!
- **Observations:** $Y_t \in \mathbb{R}^m$
  - Sensor readings at time t.
  - Noisy, partial glimpses of the hidden state.
- **Goal:** Given observations up to time t, i.e. $Y_{1:t}$,
  - Estimate the current state X_t.
  - Sometimes even forecast future states $X_{t+1}, X_{t+2}, \ldots$

Key Idea

SSMs capture how the world evolves (*state transition*) and how we perceive it (*observation model*).

Bayesian Filtering: The Big Picture

Our Goal: Maintain a belief about the hidden state X_t as we receive observations $Y_{1:t}$.

Recursive Bayesian Estimation

At every time step, we perform two key operations:

1. **Prediction (Time Update):**

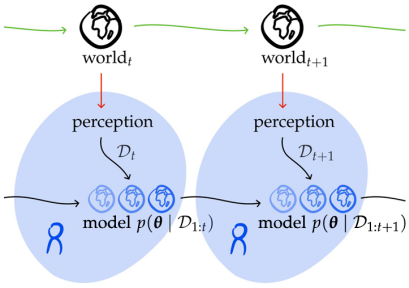
$$p(X_t \mid Y_{1:t-1}) = \int p(X_t \mid X_{t-1})\, p(X_{t-1} \mid Y_{1:t-1})\, dX_{t-1}$$

2. **Update (Measurement Correction):**

$$p(X_t \mid Y_{1:t}) \propto p(Y_t \mid X_t)\, p(X_t \mid Y_{1:t-1})$$

Repeat: These steps are performed in a loop as new data arrives.

Figure 3.1: How Bayesian Filtering Works



Explanation:

- The **true world state** evolves over time: world$_t \rightarrow$ world$_{t+1}$.
- The agent perceives each state through noisy data: D_t, D_{t+1}.
- Based on its observations $D_{1:t}$, it maintains a belief (posterior) over the hidden variables:

$$p(\theta \mid D_{1:t})$$

- At each time step, the agent:
  1. **Updates** its belief using new data.
  2. **Predicts** the next state of the world.

Core Idea

Filtering is the cycle of *observe* $\rightarrow$*update belief* $\rightarrow$*predict* $\rightarrow$*repeat*.

Introducing the Kalman Filter

Kalman Filter: A powerful, tractable solution to the filtering problem — when everything is linear and Gaussian.

Assumptions

- Initial state: Gaussian prior over X_0
- Dynamics model: Linear with Gaussian process noise
- Observation model: Linear with Gaussian measurement noise

Why it's useful:

- The belief at each step — the posterior $p(X_t \mid Y_{1:t})$ — stays Gaussian.
- All filtering steps (prediction and update) have closed-form analytical solutions.

Key Insight

Linear-Gaussian systems allow exact, recursive inference — no sampling or approximations needed.

Kalman Filter: Formal Definition

The Kalman Filter is defined by three components:

1. **Initial State (Prior Belief):**

$$X_0 \sim \mathcal{N}(\mu_0, \Sigma_0) \tag{3.1}$$

2. **State Transition Model (Dynamics):**

$$X_{t+1} = FX_t + \varepsilon_t, \quad \varepsilon_t \sim \mathcal{N}(0, \Sigma_x) \tag{3.2}$$

   - F: State transition matrix ($d \times d$)
   - ε_t: Process noise (may include drift; zero-mean assumed here)

3. **Observation Model (Sensor):**

$$Y_t = HX_t + \eta_t, \quad \eta_t \sim \mathcal{N}(0, \Sigma_y) \tag{3.3}$$

   - H: Observation matrix ($m \times d$)
   - η_t: Measurement noise

Note: F, H, Σ_x, Σ_y are assumed to be known (can be time-varying).

Graphical Model and Conditional Independencies

Kalman Filter as a Probabilistic Graphical Model:

- Hidden state sequence: $X_1 \to X_2 \to X_3 \to \dots$
- Each X_t emits an observation Y_t

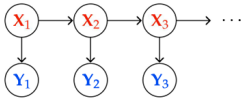


Figure 3.2: Directed graphical model of a Kalman filter with hidden states $\mathbf{X}_t$ and observables $\mathbf{Y}_t$.

Conditional Independencies

- **Markov Property (Eq. 3.4):**

$$X_{t+1} \perp\!\!\!\perp X_{1:t-1}, Y_{1:t-1} \mid X_t$$

Future state depends only on the current state.

Graphical Model and Conditional Independencies

Conditional Independencies

- **Observation Dependence (Eq. 3.5):**

$$Y_t \perp\!\!\!\perp X_{1:t-1} \mid X_t$$

  Observation depends only on the current state.

- **Observation Independence (Eq. 3.6):**

$$Y_t \perp\!\!\!\perp Y_{1:t-1} \mid X_{t-1}$$

  Given the past state, current observation is independent of past observations.

The Goal: Online Recursive Estimation

What we want:

- At each time t, compute the belief:

$$p(X_t \mid Y_{1:t})$$

- Without re-processing all past data from scratch!

Why Online?

- Observations arrive sequentially in real-time.
- We need to update our estimate efficiently and recursively.
- No need to store all $Y_{1:t}$ — just the current posterior!

Recursive Filtering Loop

1. Predict: Use dynamics model to forecast next state.
2. Update: Incorporate new observation.
3. Repeat...

This is the essence of recursive Bayesian estimation.

Algorithm: Bayesian Filtering

Algorithm 3.2: Bayesian filtering

start with a prior over initial states $p(\boldsymbol{x}_0)$

for $t = 1$ **to** ∞ **do**

  assume we have $p(\boldsymbol{x}_t \mid \boldsymbol{y}_{1:t-1})$

  conditioning: compute $p(\boldsymbol{x}_t \mid \boldsymbol{y}_{1:t})$ using the new observation $\boldsymbol{y}_t$

  prediction: compute $p(\boldsymbol{x}_{t+1} \mid \boldsymbol{y}_{1:t})$

Bayesian Filtering: Prediction and Conditioning

1. Conditioning (Update Step)

Incorporate new observation Y_t to refine belief about X_t

$$p(X_t \mid Y_{1:t}) \propto p(Y_t \mid X_t) \cdot p(X_t \mid Y_{1:t-1}) \tag{3.8}$$

- Combines the prior (predicted) belief with the new observation.
- Uses Bayes' Rule.

Bayesian Filtering: Prediction and Conditioning

2. Prediction (Time Update Step)

Project the belief forward to estimate X_{t+1}

$$p(X_{t+1} \mid Y_{1:t}) = \int p(X_{t+1} \mid X_t) \cdot p(X_t \mid Y_{1:t})\, dX_t \tag{3.9}$$

- Uses the transition model to forecast the next state.
- Allows filtering to continue over time.

Deriving the Conditioning (Update) Step

Goal: Update belief about current state given a new observation:

$$p(X_t \mid Y_{1:t}) = \text{posterior}$$

Start from:

$$p(X_t \mid Y_{1:t}) = p(X_t \mid Y_t, Y_{1:t-1})$$

Apply Bayes' Rule:

$$p(X_t \mid Y_{1:t}) \propto p(Y_t \mid X_t, Y_{1:t-1}) \cdot p(X_t \mid Y_{1:t-1})$$

Use Conditional Independence (Eq. 3.6):

$$Y_t \perp Y_{1:t-1}, X_{1:t-1} \setminus X_t \mid X_t \Rightarrow p(Y_t \mid X_t, Y_{1:t-1}) = p(Y_t \mid X_t)$$

Final form:

$$p(X_t \mid Y_{1:t}) \propto p(Y_t \mid X_t) \cdot p(X_t \mid Y_{1:t-1}) \tag{3.8}$$

Interpretation

Posterior $\propto$ Likelihood $\times$ Prior

Deriving the Prediction Step

Goal: Predict the next state before the next observation arrives:

$$p(X_{t+1} \mid Y_{1:t})$$

Step 1: Apply marginalization

$$p(X_{t+1} \mid Y_{1:t}) = \int p(X_{t+1}, X_t \mid Y_{1:t})\, dX_t$$

Step 2: Use the product rule

$$= \int p(X_{t+1} \mid X_t, Y_{1:t}) \cdot p(X_t \mid Y_{1:t})\, dX_t$$

Step 3: Apply the Markov Property (Eq. 3.4):

$$X_{t+1} \perp Y_{1:t} \mid X_t \Rightarrow p(X_{t+1} \mid X_t, Y_{1:t}) = p(X_{t+1} \mid X_t)$$

Final Result:

$$p(X_{t+1} \mid Y_{1:t}) = \int p(X_{t+1} \mid X_t) \cdot p(X_t \mid Y_{1:t})\, dX_t \tag{3.9}$$

Kalman Filter: The Prediction Step

Given: Posterior at time t:

$$X_t \mid Y_{1:t} \sim \mathcal{N}(\mu_t, \Sigma_t)$$

We want: Predictive belief for time $t+1$:

$$X_{t+1} \mid Y_{1:t} \sim \mathcal{N}(\hat{\mu}_{t+1}, \hat{\Sigma}_{t+1})$$

Apply the transition model:

$$X_{t+1} = FX_t + \varepsilon_t, \quad \varepsilon_t \sim \mathcal{N}(0, \Sigma_x)$$

Predicted Mean (Eq. 3.20)

$$\hat{\mu}_{t+1} = F\mu_t$$

Predicted Covariance (Eq. 3.21)

$$\hat{\Sigma}_{t+1} = F\Sigma_t F^\top + \Sigma_x$$

Kalman Filter: The Update Step

We have:

- Prior prediction: $X_{t+1} \mid Y_{1:t} \sim \mathcal{N}(\hat{\mu}_{t+1}, \hat{\Sigma}_{t+1})$
- New observation: $Y_{t+1} = HX_{t+1} + \eta_{t+1},\ \eta_{t+1} \sim \mathcal{N}(0, \Sigma_y)$

Kalman Gain (Eq. 3.18d)

$$K_{t+1} = \hat{\Sigma}_{t+1} H^\top \left(H \hat{\Sigma}_{t+1} H^\top + \Sigma_y \right)^{-1}$$

Updated Mean (Eq. 3.18b)

$$\mu_{t+1} = \hat{\mu}_{t+1} + K_{t+1} \left(Y_{t+1} - H \hat{\mu}_{t+1} \right)$$

Updated Covariance (Eq. 3.18c)

$$\Sigma_{t+1} = \left(I - K_{t+1} H \right) \hat{\Sigma}_{t+1}$$

Example 3.4 – Random Walk in 1D

System Setup:

- **State:** $X_t \in \mathbb{R}$ (position on a line)
- **Motion Model:**

$$X_{t+1} = X_t + \varepsilon_t, \quad \varepsilon_t \sim \mathcal{N}(0, \sigma_x^2)$$

- **Sensor Model:**

$$Y_t = X_t + \eta_t, \quad \eta_t \sim \mathcal{N}(0, \sigma_y^2)$$

- **Initial Belief:** $X_0 \sim \mathcal{N}(\mu_0, \sigma_0^2)$

Kalman Filter simplifies to 1D:

- Prediction and update equations become scalar operations.
- Still recursive!
- Posterior: $X_t \mid Y_{1:t} \sim \mathcal{N}(\mu_t, \sigma_t^2)$

Key Idea

Even this simple setup captures the trade-off between motion uncertainty and noisy observations.

Random Walk – 1D Kalman Update (Eq. 3.13)

Recall:

- Prior at time t: $X_t \mid Y_{1:t} \sim \mathcal{N}(\mu_t, \sigma_t^2)$
- Prediction for $t+1$:

$$X_{t+1} \mid Y_{1:t} \sim \mathcal{N}(\mu_{t+1}, \sigma_{t+1}^2)$$

- Observation: $Y_{t+1} = X_{t+1} + \eta_{t+1},\ \eta_{t+1} \sim \mathcal{N}(0, \sigma_y^2)$

Update step (Eq. 3.13):

$$\mu_{t+1} = \frac{\sigma_y^2 \mu_t + (\sigma_t^2 + \sigma_x^2) Y_{t+1}}{\sigma_t^2 + \sigma_x^2 + \sigma_y^2}$$

$$\sigma_{t+1}^2 = \frac{(\sigma_t^2 + \sigma_x^2)\sigma_y^2}{\sigma_t^2 + \sigma_x^2 + \sigma_y^2}$$

Kalman Gain λ and Equivalent Update Forms

Kalman Gain in 1D (Eq. 3.14):

$$\lambda = \frac{\sigma_t^2 + \sigma_x^2}{\sigma_t^2 + \sigma_x^2 + \sigma_y^2}$$

Updated Mean (Eq. 3.15, 3.16):

$$\mu_{t+1} = (1 - \lambda)\mu_t + \lambda Y_{t+1} \tag{3.15}$$

$$= \mu_t + \lambda(Y_{t+1} - \mu_t) \tag{3.16}$$

Updated Variance (Eq. 3.17):

$$\sigma_{t+1}^2 = (1 - \lambda)(\sigma_t^2 + \sigma_x^2) = \lambda\sigma_y^2$$

Bayesian Linear Regression as a Kalman Filter

Problem Setup:

- We want to estimate a fixed but unknown weight vector $w^* \in \mathbb{R}^d$
- Observations arrive sequentially: (x_t, y_t), where

$$y_t = x_t^\top w^* + \eta_t, \quad \eta_t \sim \mathcal{N}(0, \sigma_n^2)$$

Kalman Filter Interpretation:

- Hidden state: $X_t = w^*$ (constant!)
- Transition model: $X_{t+1} = X_t + 0 \rightarrow$no process noise $\rightarrow \Sigma_x = 0$
- Sensor model:

$$Y_t = x_t^\top X_t + \eta_t$$

- Observation matrix: $H_t = x_t^\top$

Connection

Online Bayesian Linear Regression is Kalman Filtering with constant hidden state.

Bayesian Smoothing: LookingBackWithMoreInfo

Filtering: Estimates the current state using data **up to** time t

$$p(X_k \mid Y_{1:k}) \quad \text{(for } k = t\text{)}$$

Smoothing: Estimates a **past** state X_k using all data **up to a later time** t

$$p(X_k \mid Y_{1:t}) \quad \text{where } k < t$$

Why Smooth?

- Gives more accurate estimate of X_k using future evidence.
- Useful in offline settings (e.g., analyzing recorded sensor data).

Bayesian Smoothing: LookingBackWithMoreInfo

Key Equation (Eq. 3.10):

$$p(x_k \mid y_{1:t}) \propto p(x_k \mid y_{1:k}) \cdot p(y_{k+1:t} \mid x_k)$$

- First term: filtering result
- Second term: likelihood of future observations
- Smoothing = Filtering + Backward Pass
- We combine past estimate with future information for better accuracy

Kalman Smoothing: Fwd-Bkwd Estimation

Filtering: Real-time estimate

$$p(X_k \mid Y_{1:k}) = \text{filtering posterior}$$

Smoothing: Offline refinement

$$p(X_k \mid Y_{1:t}) = \text{smoothing posterior}, \quad k < t$$

Linear-Gaussian Case:

- Both filtering and smoothing posteriors are Gaussian.
- Can compute smoothing efficiently using a backward recursion.

Kalman Smoothing: Fwd-Bkwd Estimation

Key Equation (Eq. 3.11):

$$p(x_k \mid y_{1:t}) = \mathcal{N}(\tilde{\mu}_k, \tilde{\Sigma}_k)$$

- Based on forward-filtered posterior $p(x_k \mid y_{1:k})$
- And the backward Kalman gain, derived from transition model

Smoothing Algorithm = Filter Forward + Smooth Backward

Also called the Forward-Backward or Two-Filter smoother.

Thank you!

THE UNIVERSITY OF NEW MEXICO

OPTIMIZATION AND ESTIMATION LAB

ONE.UNM.EDU