APMA4903 Research Project

Application of Kalman Filter in Finance

Bella Wu (bw2604@columbia.edu)
Maggie Yang (my2652@columbia.edu)
Junhui Zhang (jz2903@columbia.edu)

Supervised by Professor Chris Wiggins

Oct 5th, 2020

References

[1] *Self-driving car object tracking: Intuition and the math behind kalman filter*. https://medium.com/@jonathan_hui/self-driving-object-tracking-intuition-and-the-math-behind-kalman-filter-657d11dd0a90.

[2] *Tutorial: The kalman filter*. http://web.mit.edu/kirtley/kirtley/binlustuff/literature/control/Kalman

[3] *Understanding kalman filters*. https://www.mathworks.com/videos/series/understanding-kalman-filters.html.

[4] *Use of kalman filter in finance and estimation of value-at-risk*. https://www.adrian.idv.hk/2019-08-18-kalman/.

[5] M. Laaraiedh, *Implementation of kalman filter with python language*, 2012.

[6] K. P. Murphy, *Machine Learning: a Probabilistic Perspective*, The MIT Press, Cambridge, Massachusetts; London, England, 2012.

[7] G. Pasricha, *Kalman filter and its economic applications*, mpra paper, University Library of Munich, Germany, 2006.

[8] C. Tripp and R. D. Shachter, *Approximate kalman filter q-learning for continuous state-space mdps*, 2013.

[9] Y. Xu and G. Zhang, *Application of kalman filter in the prediction of stock price*, in Proc International Symposium on Knowledge Acquisition and Modeling (KAM), 2015.

- About us
- Why this topic?
- Why we care?

Content

- Part One: Past
  - History of Kalman Filter (Bella)
  - Idea of Kalman Filter (Bella)
  - Intuitive Example: Car Tracking (Bella)
  - Mathematical Derivation (Junhui)
  - Algorithm (Maggie)
  - Code Example: tracking a mobile in wireless network (Maggie)
- Part Two: Now
  - Application in Financial field (Junhui)
  - Financial Example: stock price prediction (Junhui)
- Part Three: Future
  - Kalman Filter and Reinforcement Learning (Maggie)
  - Future Extension (Maggie)
  - Conclusion (Maggie)

Part One

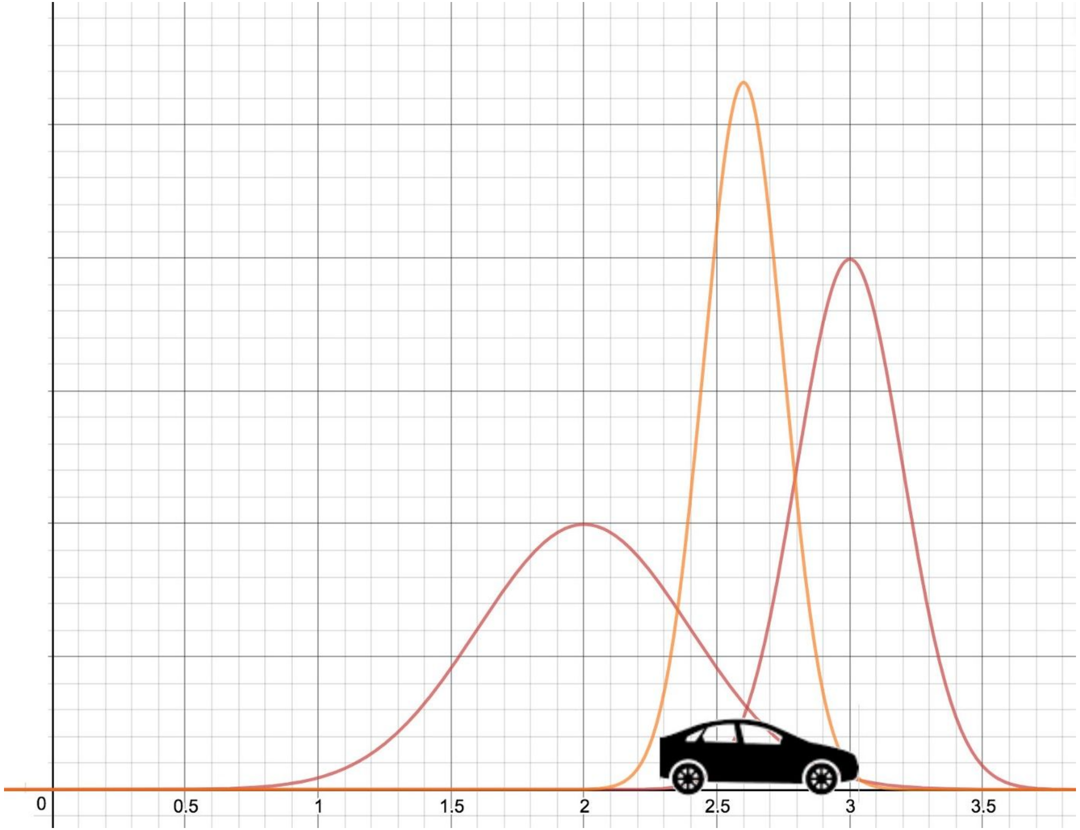
Past

History and Basic Idea

- Kalman filtering, also known as linear quadratic estimation (LQE)
- Peter Swerling (1958), Rudolf E. Kalman (1960) and Richard S. Bucy (1961)
- Application:
  - Guidance, navigation, and control of vehicles
  - Signal processing
  - Financial markets
  - Apollo project navigation
  - NASA space shuttle, navy submarines, unmanned aerospace vehicles and weapons
- Extensions and generalizations: Extended Kalman Filter and Unscented Kalman Filter

Rudolf E. Kálmán

Intuitive Example: Car Model



Belief

+

Measurement

Final Prediction

State Transition Model

State of Car

$$x_k = \begin{bmatrix} p_k \\ v_k \end{bmatrix}$$

where

p_k and v_k are the position and velocity along x-axis at $time = k$.

State of Motion

$$p_k = p_{k-1} + v_{k-1}\Delta t$$
$$v_k = v_{k-1}$$

Matrix Form

$$x_k = \begin{bmatrix} p_k \\ v_k \end{bmatrix} = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} p_{k-1} \\ v_{k-1} \end{bmatrix} = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} x_{k-1}$$

$$x_k = \mathbf{A}\, x_{k-1}$$

Input Controls

$$p_k = p_{k-1} + v_{k-1}\Delta t + \frac{1}{2}a\Delta t^2$$
$$v_k = v_{k-1} + a\Delta t$$

$$\Rightarrow$$

$$x_k = \mathbf{A}\, x_{k-1} + \mathbf{B}\, u_k$$

$$x_k = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} x_{k-1} + \begin{bmatrix} \frac{1}{2}\Delta t^2 \\ \Delta t \end{bmatrix} \begin{bmatrix} a \end{bmatrix}$$

$$\mathbf{B} = \begin{bmatrix} \frac{1}{2}\Delta t^2 \\ \Delta t \end{bmatrix}$$
$$u_k = \begin{bmatrix} a \end{bmatrix}$$

Input Controls (with process noise)

$$x_k = \mathbf{A}\, x_{k-1} + \mathbf{B}\, u_k + w_k$$

$$w_k \sim \mathcal{N}(0, Q)$$

Observer Model (Measurement Model)

$$y_k = \mathbf{C}\, x_k + v_k$$

$$v_k \sim \mathcal{N}(0, R)$$

Putting Together

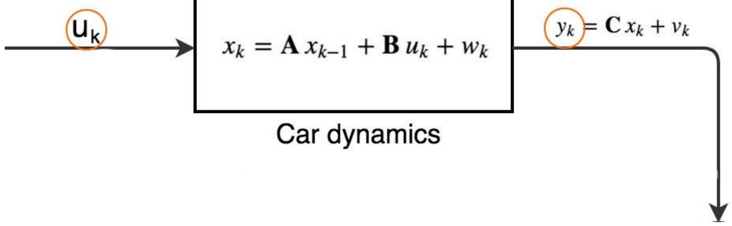
$$x_k = \mathbf{A}\, x_{k-1} + \mathbf{B}\, u_k + w_k$$
$$y_k = \mathbf{C}\, x_k + v_k$$

where

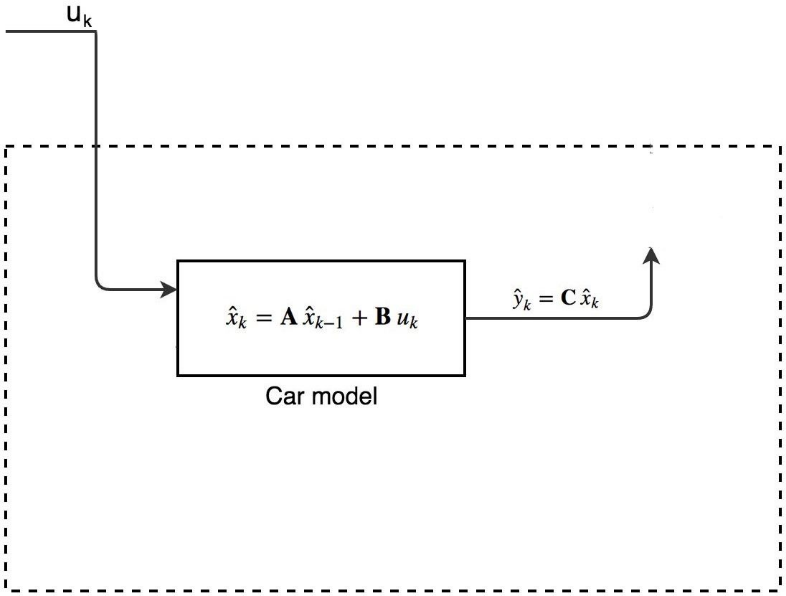
x_k and x_{k-1} are the states of the system at $time = k$ and $k-1$ respectively.
$\mathbf{A}$ is the state-transition model from state x_{k-1} to x_k.
$\mathbf{B}$ is the input-control model that applies to the control vector u_k.
$w_k \sim \mathcal{N}(0, Q_k)$ is the sampled process noise, like wind.
Q_k is the covariance matrix of the process noise.

Observation/measurement:
y_k is the measurements made at $time = k$.
$\mathbf{C}$ is the observation model to convert the state x_k to measurements.
$v_k \sim \mathcal{N}(0, R_k)$ is the sampled measurement noise like sensor noise.
R_k is the covariance matrix of the measurement noise.

Idea of Kalman Filter

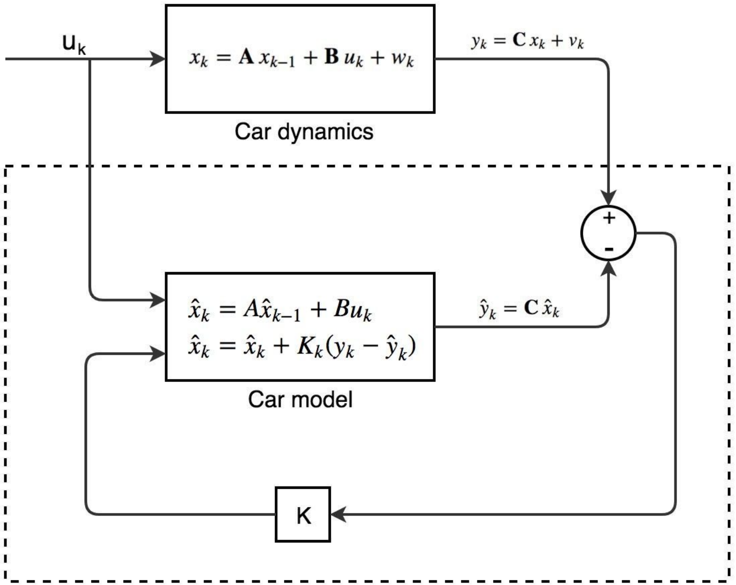


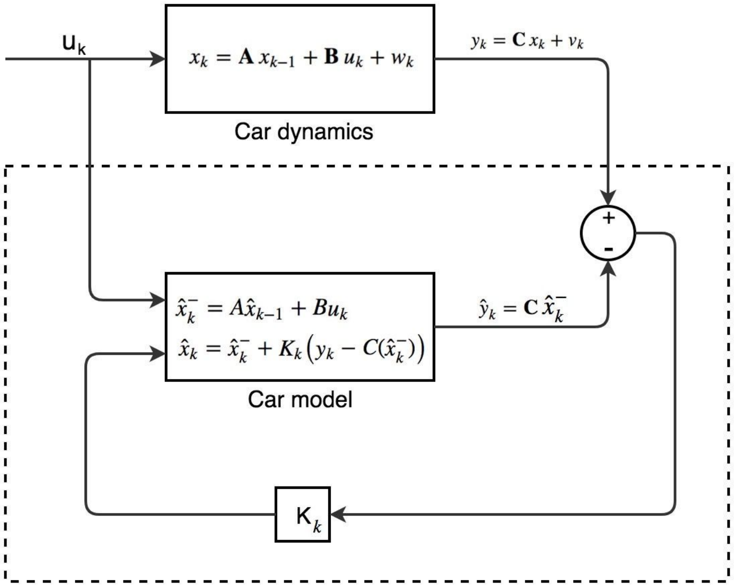
Real World



Observer

Idea of Kalman Filter





Quick Peek at Kalman Gain (K)

$$K_k = \frac{P_k^- \mathbf{C}^\mathbf{T}}{\mathbf{C} P_k^- \mathbf{C}^\mathbf{T} + R}$$

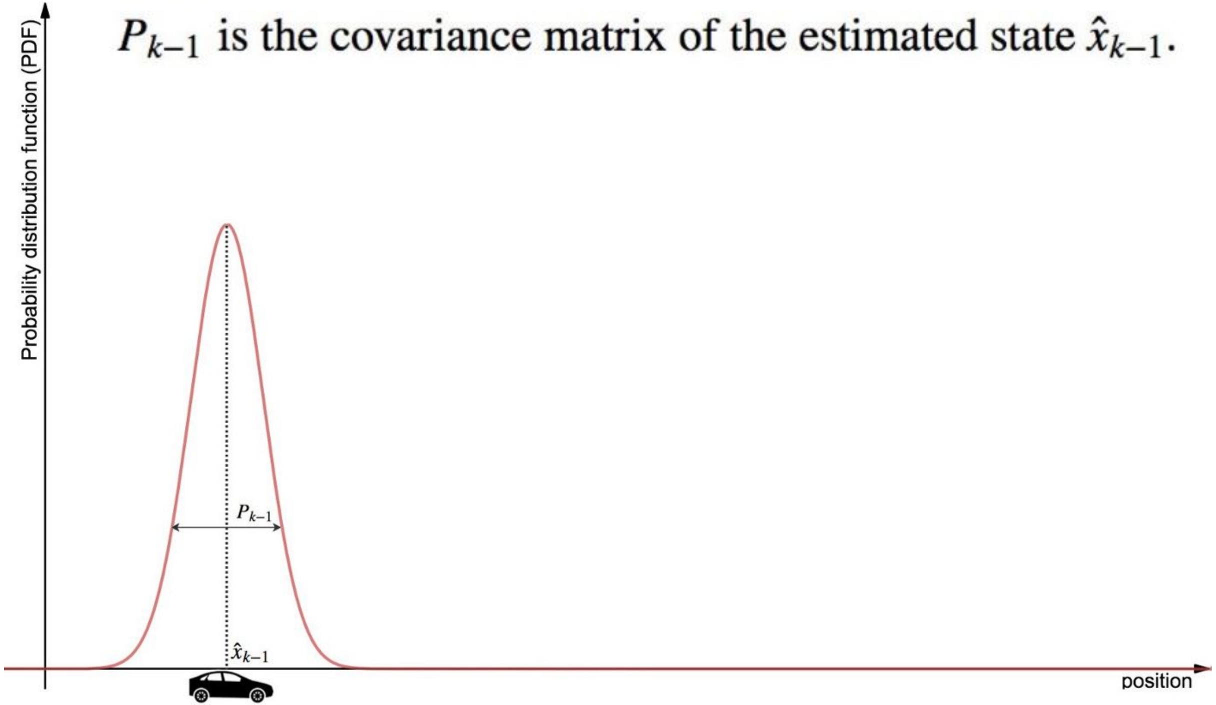
Sanity Check

$$\begin{aligned}
\hat{x}_k &= A\hat{x}_{k-1} + Bu_k + K_k(y_k - \hat{y}_k) \\
&= A\hat{x}_{k-1} + Bu_k + K_k(y_k - C(A\hat{x}_{k-1} + Bu_k)) \\
&= A\hat{x}_{k-1} + Bu_k + (y_k - A\hat{x}_{k-1} + Bu_k) \\
&= y_k
\end{aligned}$$

Prediction

$\hat{x}_{k-1}$ is the estimated state at $time = k - 1$.

P_{k-1} is the covariance matrix of the estimated state $\hat{x}_{k-1}$.

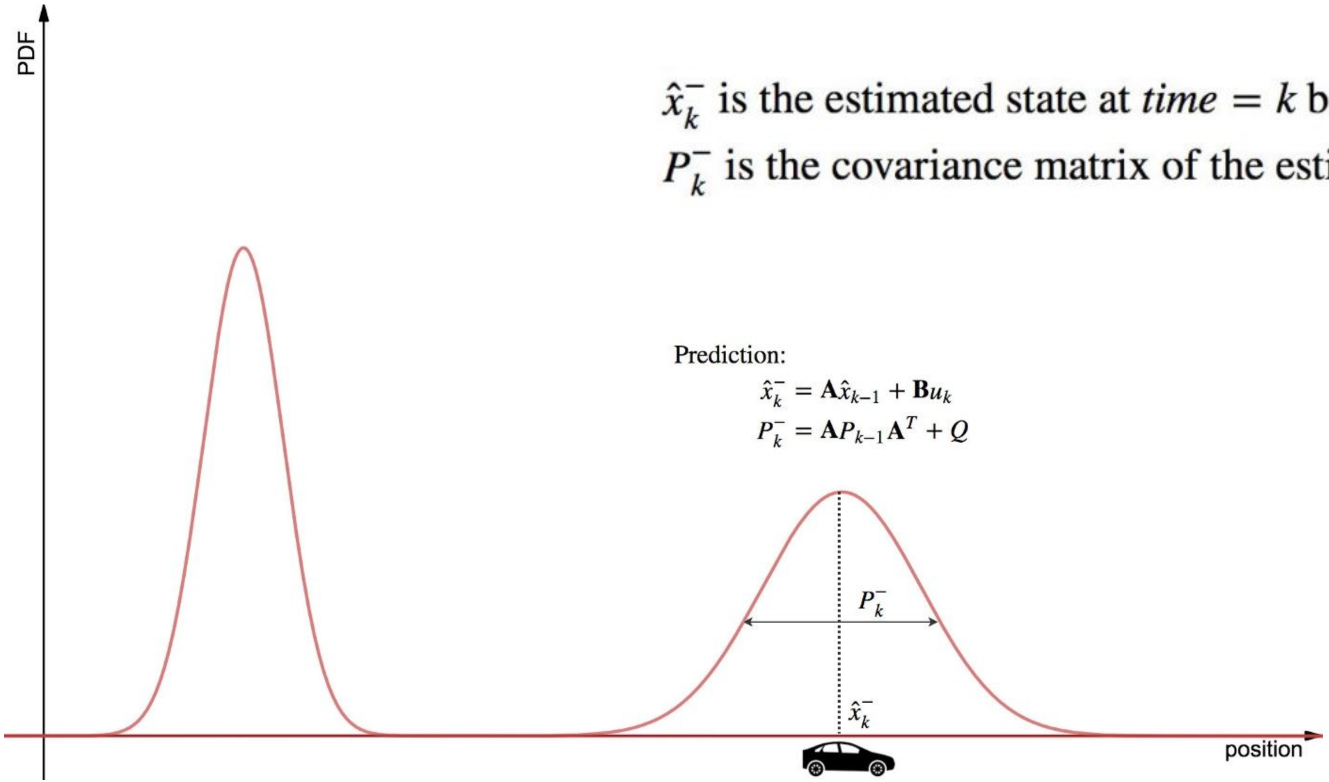


$$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_k$$

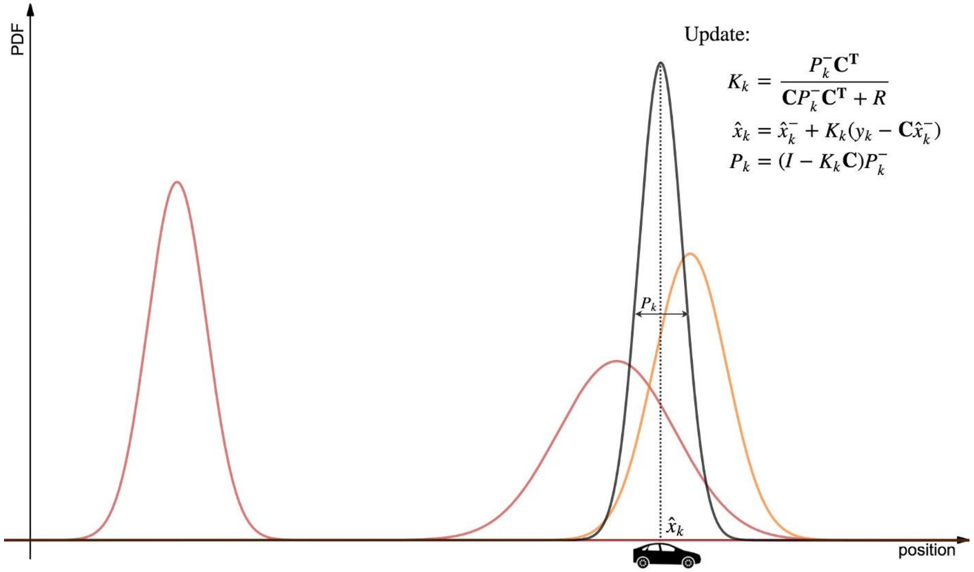
Prediction

$\hat{x}_k^-$ is the estimated state at $time = k$ before the update.

P_k^- is the covariance matrix of the estimated state $\hat{x}_k^-$.



Updated State Estimation



K_k is the computed Kalman gain to correct the observer estimation.

$\hat{x}_k$ is the estimated state at $time = k$.

P_k is the covariance matrix of the estimated state $\hat{x}_k$.

Recap

Prediction

$$\hat{x}_k^- = \mathbf{A}\hat{x}_{k-1} + \mathbf{B}u_k$$
$$P_k^- = \mathbf{A}P_{k-1}\mathbf{A}^T + Q$$

Update

$$K_k = \frac{P_k^-\mathbf{C}^\mathbf{T}}{\mathbf{C}P_k^-\mathbf{C}^\mathbf{T} + R}$$
$$\hat{x}_k = \hat{x}_k^- + K_k(y_k - \mathbf{C}\hat{x}_k^-)$$
$$P_k = (I - K_k\mathbf{C})P_k^-$$

where

$\hat{x}_{k-1}$ is the estimated state at $time = k - 1$.

P_{k-1} is the covariance matrix of the estimated state $\hat{x}_{k-1}$.

u_k is the input control.

$\mathbf{A}$ is the state-transition model.

$\mathbf{B}$ is the input-control model.

$\mathbf{C}$ is the observer model for the measurement.

Q is the covariance matrix of the process noise.

R is the covariance matrix of the measurement noise.

y_k is the measurement at $time = k$.

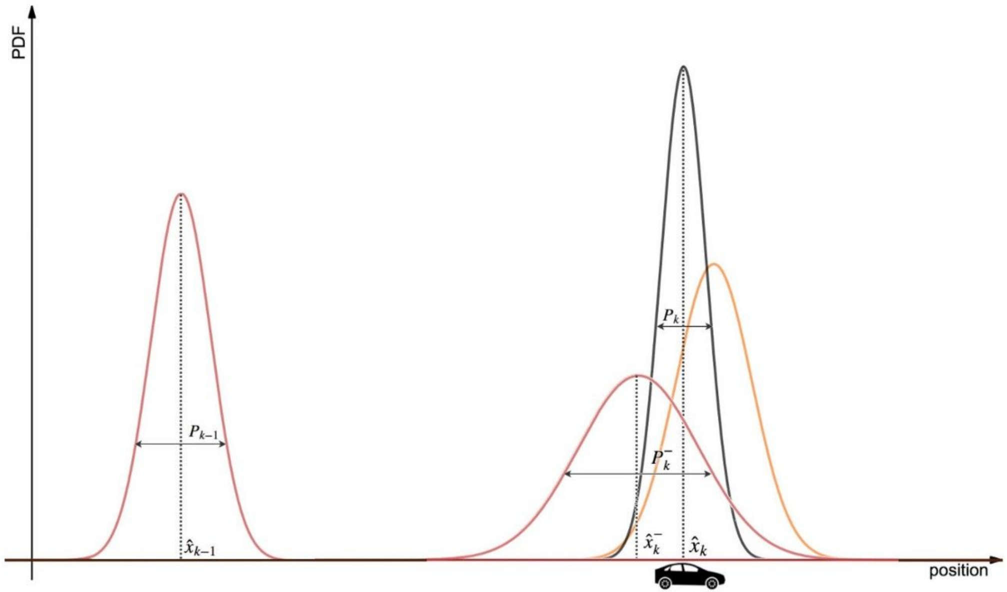
$\hat{x}_k^-$ is the estimated state at $time = k$ before the update.

P_k^- is the covariance matrix of the estimated state $\hat{x}_k^-$.

K_k is the computed Kalman gain to correct the observer estimation.

$\hat{x}_k$ is the estimated state at $time = k$.

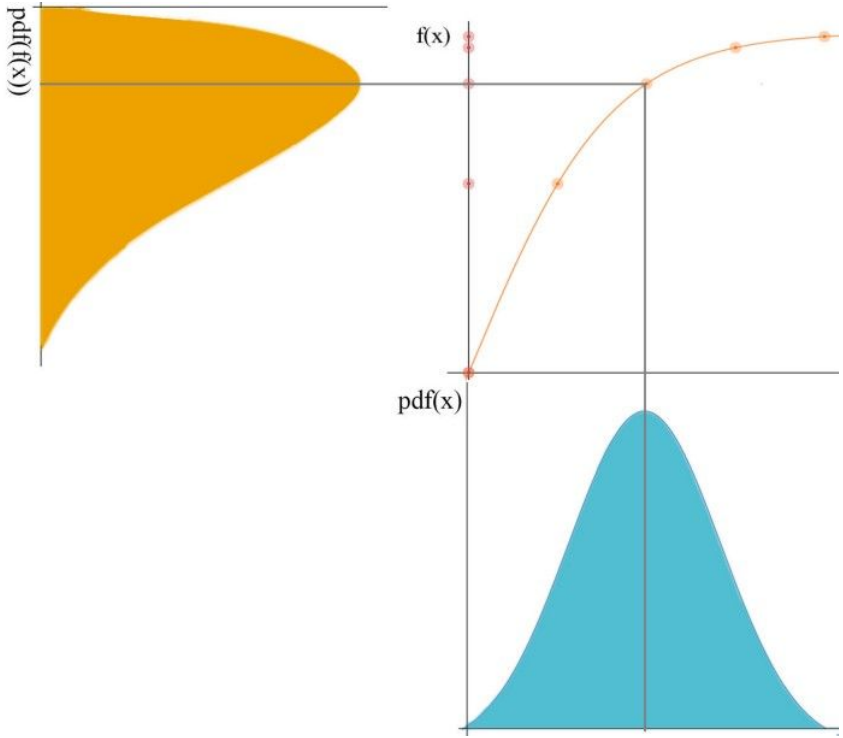
P_k is the covariance matrix of the estimated state $\hat{x}_k$.



Extended Kalman Filter

$$x_k = A\,x_{k-1} + B\,u_k + w_k$$
$$y_k = C\,x_k + v_k$$

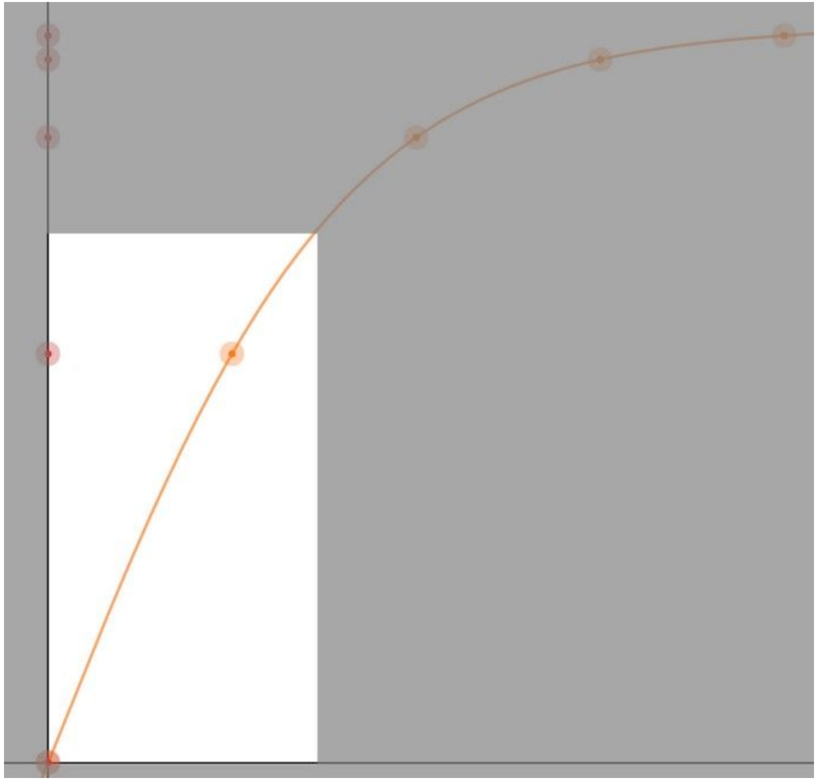
$$x_k = f(x_{k-1}, u_k) + w_k$$
$$y_k = h(x_k) + v_k$$



Extended Kalman Filter

$$x_k = A\, x_{k-1} + B\, u_k + w_k$$
$$y_k = C\, x_k + v_k$$

$$x_k = f(x_{k-1}, u_k) + w_k$$
$$y_k = h(x_k) + v_k$$



Extended Kalman Filter

Prediction:

$$\hat{x}_k^- = \mathbf{A}\hat{x}_{k-1} + \mathbf{B}u_k$$
$$P_k^- = \mathbf{A}P_{k-1}\mathbf{A}^T + Q$$

Update:

$$K_k = \frac{P_k^-\mathbf{C}^\mathbf{T}}{\mathbf{C}P_k^-\mathbf{C}^\mathbf{T} + R}$$
$$\hat{x}_k = \hat{x}_k^- + K_k(y_k - \mathbf{C}\hat{x}_k^-)$$
$$P_k = (I - K_k\mathbf{C})P_k^-$$

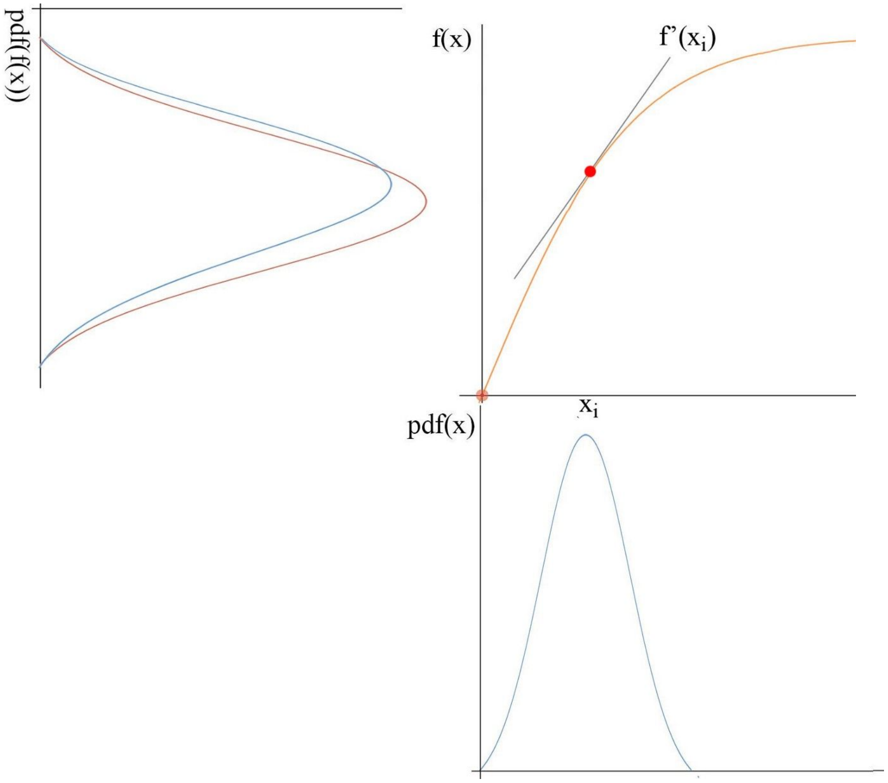
Prediction:

$$\hat{x}_k^- = f(\hat{x}_{k-1}, u_k)$$
$$P_k^- = \mathbf{F}P_{k-1}\mathbf{F}^T + Q$$
$$\mathbf{F} = \left.\frac{\partial f}{\partial x}\right|_{\hat{x}_{k-1},u_k}$$

Update:

$$K_k = \frac{P_k^- H^T}{\mathbf{H}P_k^-\mathbf{H}^\mathbf{T} + R}$$
$$\hat{x}_k = \hat{x}_k^- + K_k(y_k - h(\hat{x}_k^-))$$
$$P_k = (I - K_k\mathbf{H})P_k^-$$
$$\mathbf{H} = \left.\frac{\partial h}{\partial x}\right|_{\hat{x}_k^-}$$

Extended Kalman Filter



Prediction:

$$\hat{x}_k^- = f(\hat{x}_{k-1}, u_k)$$

$$P_k^- = \mathbf{F} P_{k-1} \mathbf{F}^T + Q$$

$$\mathbf{F} = \left.\frac{\partial f}{\partial x}\right|_{\hat{x}_{k-1}, u_k}$$

Update:

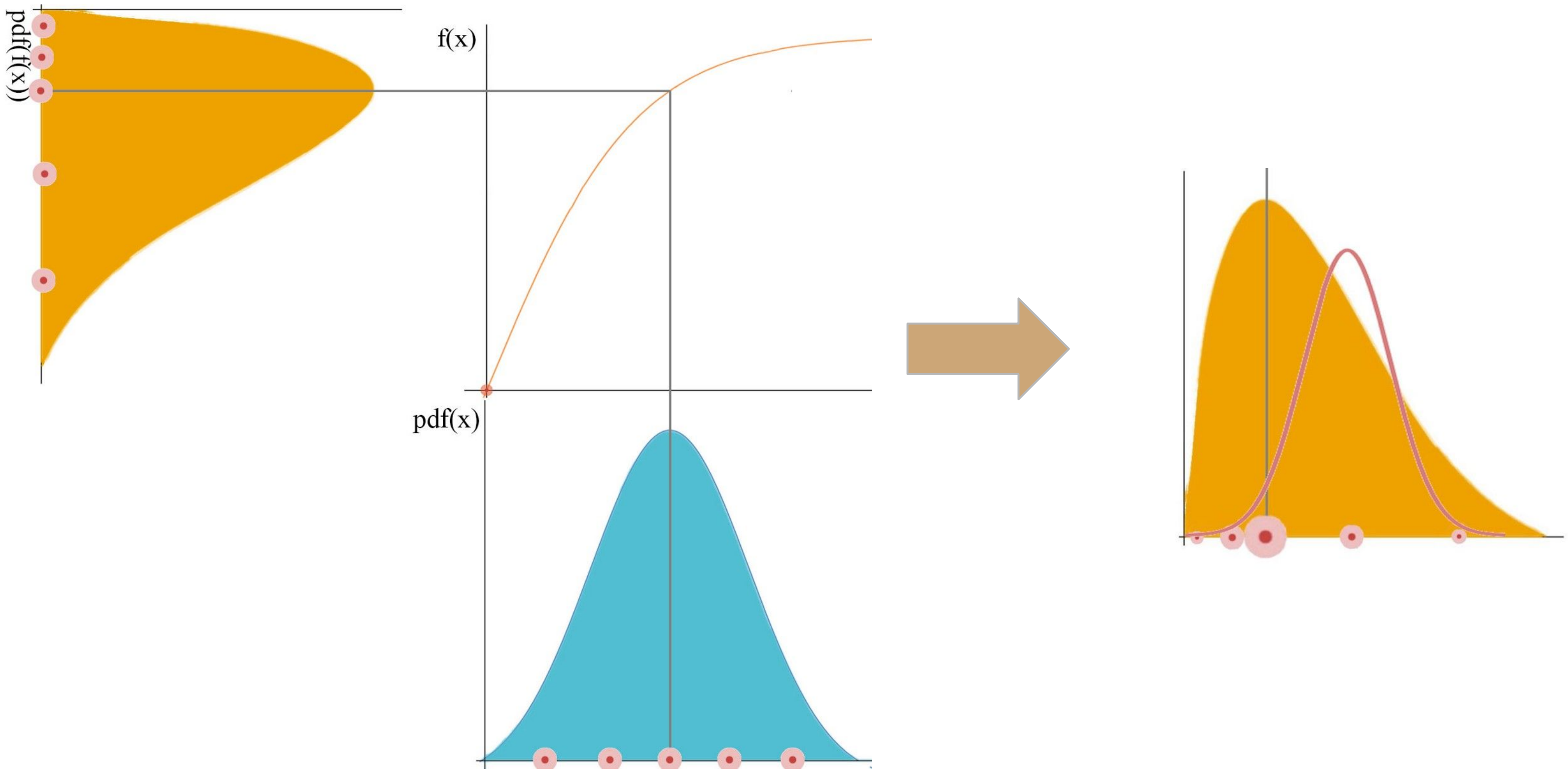
$$K_k = \frac{P_k^- H^T}{\mathbf{H} P_k^- \mathbf{H}^\mathbf{T} + R}$$

$$\hat{x}_k = \hat{x}_k^- + K_k(y_k - h(\hat{x}_k^-))$$

$$P_k = (I - K_k \mathbf{H}) P_k^-$$

$$\mathbf{H} = \left.\frac{\partial h}{\partial x}\right|_{\hat{x}_k^-}$$

Unscented Kalman Filter



Derivation 1

The Kalman Filter is the ***optimal MSE filter****.*

Note added by CW: beware a change of notation here:
y->z , C->H, A->Phi, ….

Derivation 1

$$
\begin{aligned}
x_{k+1} &= \Phi x_k + w_k & \quad Q &= E\left[w_k w_k^T\right] \\
z_k &= H x_k + v_k & \quad R &= E\left[v_k v_k^T\right]
\end{aligned}
$$

Derivation 1

$$
\begin{aligned}
x_{k+1} &= \Phi x_k + w_k & Q &= E\left[w_k w_k^T\right] \\
z_k &= H x_k + v_k & R &= E\left[v_k v_k^T\right]
\end{aligned}
$$

$$\hat{x}'_{k+1} = \Phi \hat{x}_k$$

Derivation 1

$$x_{k+1} = \Phi x_k + w_k \qquad Q = E\left[w_k w_k^T\right]$$

$$z_k = H x_k + v_k \qquad R = E\left[v_k v_k^T\right]$$

$$\hat{x}'_{k+1} = \Phi \hat{x}_k$$

$$\hat{x}_k = \hat{x}'_k + K_k\left(z_k - H\hat{x}'_k\right)$$

$$\hat{x}_k = \hat{x}'_k + K_k\left(H x_k + v_k - H\hat{x}'_k\right)$$

Derivation 1

$$\hat{x}_k = \hat{x}'_k + K_k(Hx_k + v_k - H\hat{x}'_k)$$

$$P_k = E\left[e_k e_k^T\right] = E\left[(x_k - \hat{x}_k)(x_k - \hat{x}_k)^T\right]$$

Derivation 1

$$\hat{x}_k = \hat{x}'_k + K_k \left(H x_k + v_k - H \hat{x}'_k \right)$$

$$P_k = E\left[e_k e_k^T\right] = E\left[\left(x_k - \hat{x}_k\right)\left(x_k - \hat{x}_k\right)^T\right]$$

$$\begin{aligned} P_k &= (I - K_k H)\, E\left[\left(x_k - \hat{x}'_k\right)\left(x_k - \hat{x}'_k\right)^T\right](I - K_k H) \\ &+ K_k E\left[v_k v_k^T\right] K_k^T \end{aligned}$$

Derivation 1

$$\hat{x}_k = \hat{x}'_k + K_k \left(H x_k + v_k - H \hat{x}'_k \right)$$

$$P_k = E\left[e_k e_k^T \right] = E\left[\left(x_k - \hat{x}_k \right) \left(x_k - \hat{x}_k \right)^T \right]$$

$$\begin{aligned} P_k &= (I - K_k H)\, E\left[\left(x_k - \hat{x}'_k \right) \left(x_k - \hat{x}'_k \right)^T \right] (I - K_k H) \\ &+ K_k E\left[v_k v_k^T \right] K_k^T \end{aligned}$$

$$P_k = (I - K_k H)\, P'_k \, (I - K_k H)^T + K_k R K_k^T$$

$$P_k = P'_k - K_k H P'_k - P'_k H^T K_k^T + K_k \left(H P'_k H^T + R \right) K_k^T$$

Derivation 1

$$\hat{x}'_{k+1} = \Phi \hat{x}_k$$

$$\begin{aligned} e'_{k+1} &= x_{k+1} - \hat{x}'_{k+1} \\ &= (\Phi x_k + w_k) - \Phi \hat{x}_k \\ &= \Phi e_k + w_k \end{aligned}$$

$$\begin{aligned} P'_{k+1} &= E\left[e'_{k+1} e^{T\prime}_{k+1}\right] \\ &= E\left[\Phi e_k \left(\Phi e_k\right)^T\right] + E\left[w_k w_k^T\right] \\ &= \Phi P_k \Phi^T + Q \end{aligned}$$

Derivation 1

$$T\left[P_k\right] \;=\; T\left[P'_k\right] \;-\; 2T\left[K_k H P'_k\right] \;+\; T\left[K_k \left(H P'_k H^T \;+\; R\right) K_k^T\right]$$

$$\frac{dT\left[P_k\right]}{dK_k} \;=\; -\,2(H P'_k)^T \;+\; 2K_k \left(H P'_k H^T \;+\; R\right)$$

Derivation 1

$$T\left[P_k\right] \;=\; T\left[P'_k\right] \;-\; 2T\left[K_k H P'_k\right] \;+\; T\left[K_k\left(HP'_kH^T \;+\; R\right)K_k^T\right]$$

$$\frac{dT\left[P_k\right]}{dK_k} \;=\; -\,2(HP'_k)^T \;+\; 2K_k\left(HP'_kH^T \;+\; R\right)$$

$$\left(HP'_k\right)^T \;=\; K_k\left(HP'_kH^T \;+\; R\right)$$

$$K_k \;=\; P'_kH^T\left(HP'_kH^T \;+\; R\right)^{-1}$$

Derivation 1

$$T\left[P_k\right] \;=\; T\left[P'_k\right] \;-\; 2T\left[K_k H P'_k\right] \;+\; T\left[K_k\left(H P'_k H^T \;+\; R\right) K_k^T\right]$$

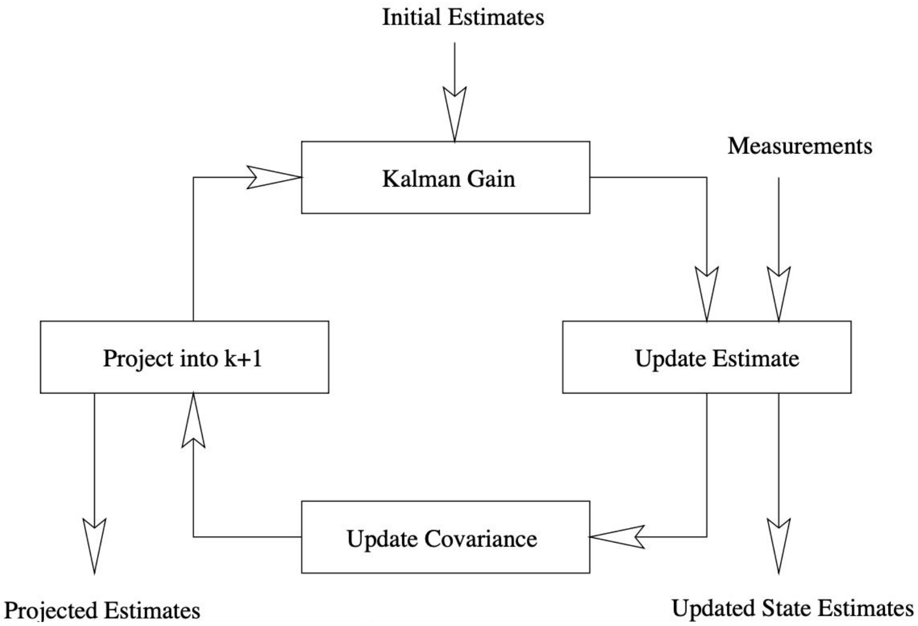
$$\frac{dT\left[P_k\right]}{dK_k} \;=\; -\,2(H P'_k)^T \;+\; 2K_k\left(H P'_k H^T \;+\; R\right)$$

$$\left(H P'_k\right)^T \;=\; K_k\left(H P'_k H^T \;+\; R\right)$$

$$K_k \;=\; P'_k H^T\left(H P'_k H^T \;+\; R\right)^{-1}$$

$$\begin{aligned} P_k &= P'_k \;-\; P'_k H^T\left(H P'_k H^T \;+\; R\right)^{-1} H P'_k \\ &= P'_k \;-\; K_k H P'_k \\ &= \left(I \;-\; K_k H\right) P'_k \end{aligned}$$

Derivation 1



Derivation 2

*The Kalman Filter is an algorithm for **exact Bayesian filtering** for linear-Gaussian state space models.*

Derivation 2

Assumptions:

$$x_{k+1} = \Phi x_k + w_k,\ E[w_k w_k^T] = Q$$

$$z_k = H x_k + v_k,\ E[v_k v_k^T] = R$$

$$p(x_k | z_{1:k}) = \mathcal{N}(x_k | \mu_k, \Sigma_k)$$

Derivation 2

Assumptions:

$$x_{k+1} = \Phi x_k + w_k,\ E[w_k w_k^T] = Q$$

$$z_k = H x_k + v_k,\ E[v_k v_k^T] = R$$

$$p(x_k | z_{1:k}) = \mathcal{N}(x_k | \mu_k, \Sigma_k)$$

Prior:

$$\begin{aligned}
p(x_k | z_{1:k-1}) &= \int \mathcal{N}(x_k | \Phi x_{k-1}, Q) \mathcal{N}(x_{k-1} | \mu_{k-1}, \Sigma_{k-1}) dx_{k-1} \\
&= \mathcal{N}(x_k | \mu_{k|k-1}, \Sigma_{k|k-1}) \\
\mu_{k|k-1} &:= \Phi \mu_{k-1} \\
\Sigma_{k|k-1} &:= \Phi \Sigma_{k-1} \Phi^T + Q
\end{aligned}$$

Derivation 2

Posterior:

$$p(x_k|z_k, z_{1:k-1}) \propto p(z_k|x_k)p(x_k|z_{1:k-1})$$

Bayes rule for Gaussian gives

$$\Sigma_k^{-1} = \Sigma_{k|k-1}^{-1} + H^T R^{-1} H$$

Applying matrix inversion lemma

$$\begin{aligned} \Sigma_k &= \Sigma_{k|k-1} - \Sigma_{k|k-1} H^T (R + H\Sigma_{k|k-1} H^T)^{-1} H \Sigma_{k|k-1} \\ &= (I - K_k H)\Sigma_{k|k-1} \end{aligned}$$

Similarly, for the mean

$$\mu_k = \Sigma_k H R^{-1} z_k + \Sigma_k \Sigma_{k|k-1}^{-1} \mu_{k|k-1} = \mu_{k|k-1} - K H^T \mu_{k|k-1}$$

Derivation 2

Corollary 4.3.1 (Matrix inversion lemma). *Consider a general partitioned matrix* $\mathbf{M} = \begin{pmatrix} \mathbf{E} & \mathbf{F} \\ \mathbf{G} & \mathbf{H} \end{pmatrix}$, *where we assume* $\mathbf{E}$ *and* $\mathbf{H}$ *are invertible. We have*

$$(\mathbf{E}-\mathbf{F}\mathbf{H}^{-1}\mathbf{G})^{-1} = \mathbf{E}^{-1}+\mathbf{E}^{-1}\mathbf{F}(\mathbf{H}-\mathbf{G}\mathbf{E}^{-1}\mathbf{F})^{-1}\mathbf{G}\mathbf{E}^{-1} \quad (4.106)$$

$$(\mathbf{E}-\mathbf{F}\mathbf{H}^{-1}\mathbf{G})^{-1}\mathbf{F}\mathbf{H}^{-1} = \mathbf{E}^{-1}\mathbf{F}(\mathbf{H}-\mathbf{G}\mathbf{E}^{-1}\mathbf{F})^{-1} \quad (4.107)$$

$$|\mathbf{E}-\mathbf{F}\mathbf{H}^{-1}\mathbf{G}| = |\mathbf{H}-\mathbf{G}\mathbf{E}^{-1}\mathbf{F}||\mathbf{H}^{-1}||\mathbf{E}| \quad (4.108)$$

Posterior:

$$p(x_k|z_k, z_{1:k-1}) \propto p(z_k|x_k)p(x_k|z_{1:k-1})$$

Bayes rule for Gaussian gives

$$\Sigma_k^{-1} = \Sigma_{k|k-1}^{-1} + H^T R^{-1} H$$

Applying matrix inversion lemma

$$\begin{aligned}\Sigma_k &= \Sigma_{k|k-1} - \Sigma_{k|k-1}H^T(R + H\Sigma_{k|k-1}H^T)^{-1}H\Sigma_{k|k-1} \\ &= (I - K_k H)\Sigma_{k|k-1}\end{aligned}$$

Similarly, for the mean

$$\mu_k = \Sigma_k H R^{-1} z_k + \Sigma_k \Sigma_{k|k-1}^{-1}\mu_{k|k-1} = \mu_{k|k-1} - K H^T \mu_{k|k-1}$$

Derivation 2

Posterior

$$p(x_k|z_{1:k}) = \mathcal{N}(x_k|\mu_k, \Sigma_k)$$
$$\mu_k := \mu_{k|k-1} + K_k r_k$$
$$\Sigma_k := (I - K_k H)\Sigma_{k|k-1}$$

where

$$r_k := z_k - \hat{z}_k$$
$$\hat{z}_k := E[z_k|z_{1:k-1}] = H\mu_{k|k-1}$$

and

$$K_k := \Sigma_{k|k-1} H^T S_k^{-1}$$

where

$$S_k = H\Sigma_{k|k-1} H^T + R$$

Algorithm

- Demo on Jupyter Notebook

Part Two

Now (application in Finance)

Application in Finance

- Time Varying Parameters in a Linear Regression
- Unobserved Components Model
- Estimating Value-at-Risk of portfolio
- Market price forecasting
- ...

Example: stock price prediction

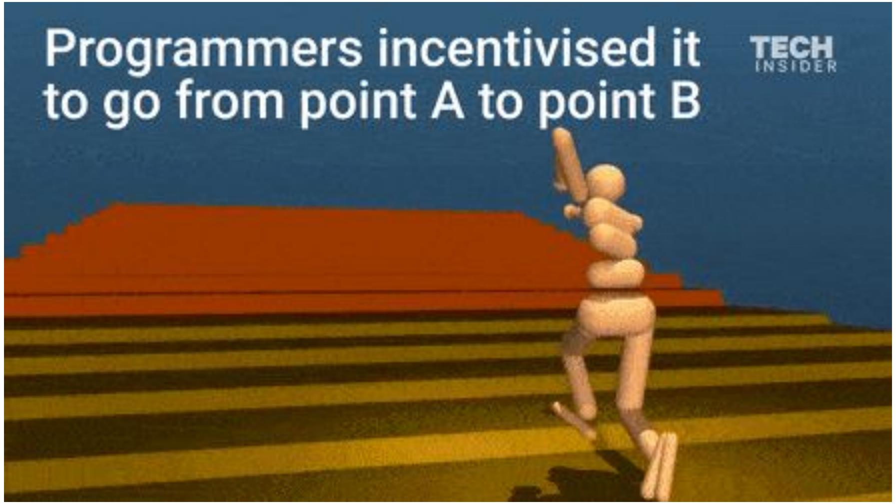
$$\begin{bmatrix} x_{k+1} \\ v_{k+1} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_k \\ v_k \end{bmatrix} + w_k$$

$$z_k = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_k \\ v_k \end{bmatrix} + v_k$$

Part Three

Future

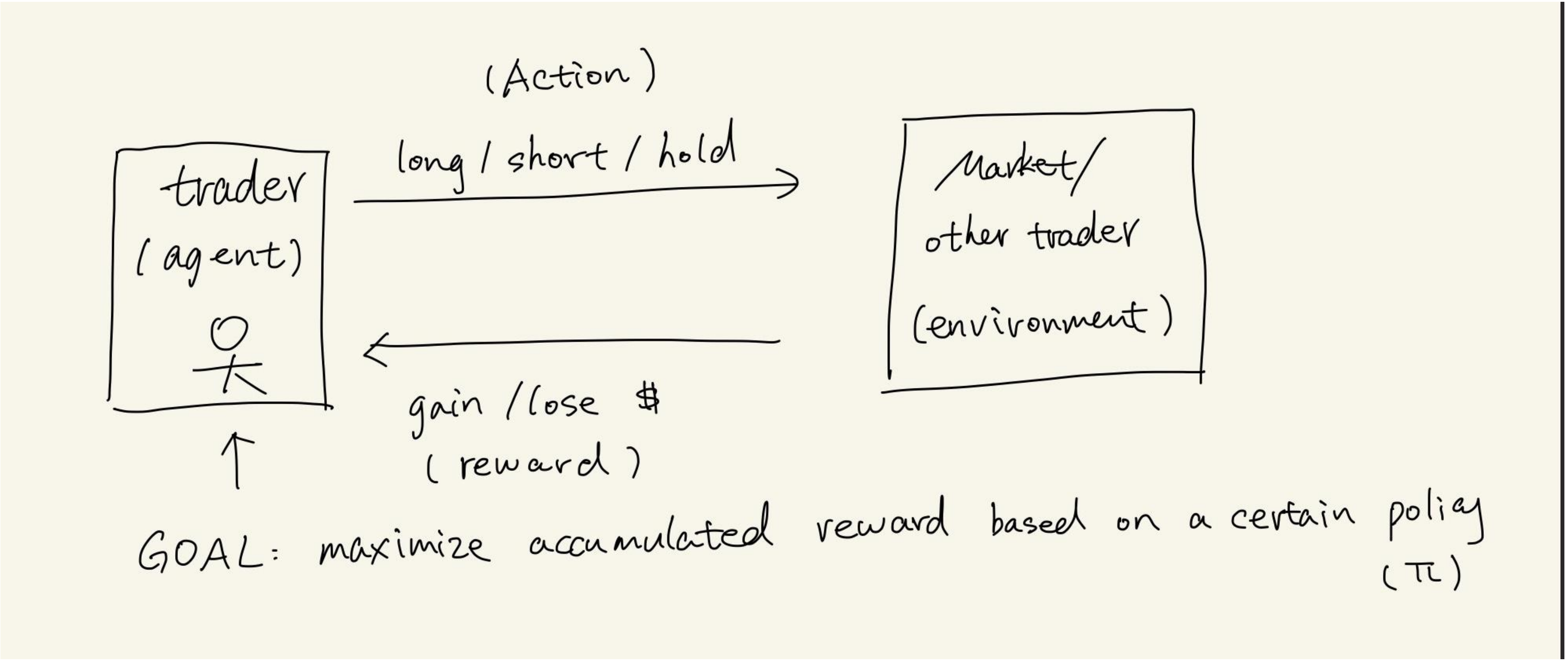
Kalman Filter and Reinforcement Learning (Q-Learning)



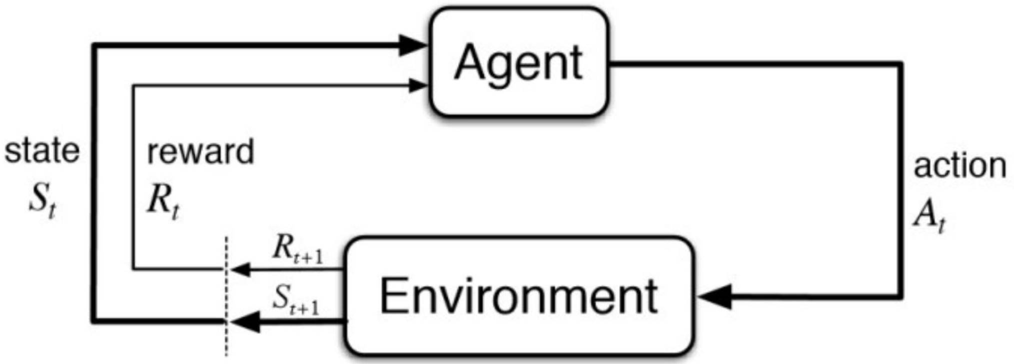
- Boston Dynamics
- DeepMind

Kalman Filter and Reinforcement Learning (Q-Learning)

Kalman Filter and Reinforcement Learning (Q-Learning)



Kalman Filter and Reinforcement Learning (Q-Learning)



Basis Reinforcement Learning is modeled as a Markov Decision Process:

- a set of environment and agent states S_t
- a set of actions A_t of the agent
- the probability of transition between states (S_t, S_{t+1}) because of an action A_t;
- the reward R_t after a transition between states (S_t, S_{t+1}) because of an action A_t
- a set of rules that describe what the agent observes

Kalman Filter and Reinforcement Learning (Q-Learning)

Ex) “Approximate Kalman Filter Q-Learning for Continuous State-Space MDPs”

Goal: Given a set of basis functions over state action pairs we search for a corresponding set of linear weights that minimizes the mean Bellman Residual

Bellman Equation:

Optimal Rewards = Maximize over first action and then follow optimal policy

$$V^*(s) = \max_a \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma V^*(s')]$$

Bellman Residual is the difference between $\|V^*(s) - V(\pi)\|$

Use Kalman Filter Q Network to optimize our policy and maximize our reward

Future Extension

- Apollo Project (1961)
- Global Positioning System (GPS)
- Unmanned Aerial Vehicle (UAV)
- Space Exploring …

- For us in the future...

Conclusion

- History
- Intuitive Idea
- Derivation
- Algorithm
- Financial Application
- Extending / Combining

Thank you!

Any Questions?