卡尔曼滤波

——基础卡尔曼滤波原理推导

杜轩

2025 年 1 月 9 日

前言

卡尔曼滤波是通过预测和测量得到最优的结果，其算法分为预测和矫正两个部分

预测：

$$\text{先验：}\hat{x_k}^- = A\hat{x_{k-1}} + Bu_{k-1}$$
$$\text{先验误差协方差：}P_k^- = AP_k^- A^T + Q$$

校正：

$$\text{卡尔曼增益：}K_k = \frac{P_k^- H^T}{HP_k^- H^T + R}$$
$$\text{后验估计：}\hat{x_k} = \hat{x_k}^- + K_k(Z_k - H\hat{x_k}^-)$$
$$\text{更新误差协方差：}P_k = (I - K_k H)P_k^-$$

杜轩

2025 年 1 月 9 日

目录

第一章　基础卡尔曼滤波原理推导 **1**

1.1　模型 1

1.2　推导 2

1.3　代码内容 5

第二章　扩展卡尔曼滤波原理推导 **9**

2.1　非线性模型 9

2.2　推导 10

第一章　基础卡尔曼滤波原理推导

1.1　模型

对于模型（状态转移方程）

$$
\begin{aligned}
x_k &= Ax_{k-1} + Bu_{k-1} + w_{k-1} \\
z_k &= Hx_k + v_k
\end{aligned}
$$

其中 u 为输入，z 是输出,A,B,H 已知，w 和 v 是干扰满足如下高斯分布

$$
\begin{aligned}
w &\sim (0, Q) \\
v &\sim (0, R)
\end{aligned}
$$

我们用 $\hat{x_k}$ 表示预测值，其中 $\hat{x_k}^-$ 代表先验估计 (通过上一步后验估计和状态转移方程计算)，$\hat{x_k}$ 代表后验估计（通过先验估计和测量得到）
在忽略 w 的噪声干扰情况下我们可以得到先验估计（就是通过已知的状态空间方程计算）则满足 $\hat{x_k}^- = A\hat{x_{k-1}} + Bu_{k-1}$
因为 x_k 测量不到所以通过 $x_k = H^{-1}z_k$ 得到忽略测量干扰

令 $\hat{x_k} = \hat{x_k}^- + R(H^{-1}z_k - \hat{x_k}^-)$，（R 属于 0 到 1），这样 R 接近零则表示预测值更信任预测，反之解决 1 更信任测量。令 $K_k = RH$ 则 $\hat{x_k} = \hat{x_k}^- + K_k(z_k - H\hat{x_k}^-)$, 其中 k_k 为卡尔曼增益

1.2 推导

我们定义误差（真实值和预测值得误差）为 $e_k = \hat{x_k} - x_k$，其中 x_k 代表真实值。我们计算误差为

$$
\begin{aligned}
e_k &= \hat{x_k} - x_k \\
&= \hat{x_k}^- + K_k(z_k - H\hat{x_k}^-) - x_k \\
&= (I - K_kH)\hat{x_k}^- + K_kz_k - x_k \\
&= (I - K_kH)\hat{x_k}^- + K_k(Hx_k + v_k) - x_k \\
&= (I - K_kH)\hat{x_k}^- - (I - K_kH)x_k + K_kv_k \\
&= (I - K_kH)(\hat{x_k}^- - x_k) + K_kv_k
\end{aligned}
$$

我们令 $e_k^- = \hat{x_k}^- - x_k$ 为先验误差

计算误差期望

$$
\begin{aligned}
E(e_k) &= E((I - K_kH)(\hat{x_k}^- - x_k)) + K_kE(v_k) \\
&= E((I - K_kH)(A\hat{x_{k-1}} + Bu_{k-1} - Ax_{k-1} - Bu_{k-1} - w_{k-1})) \\
&= (I - K_kH)AE(\hat{x_{k-1}} - x_{k-1}) - E(w_{k-1})
\end{aligned}
$$

我们假设初始误差期望为 0，即可得 $E(e_k) = 0$

我们要使得结果最优即方差最小估计我们计算 e_k 得协方差矩阵 P_k，其对角线即为方差。

$$
\begin{aligned}
P_k &= E(ee^T) \\
&= E(((I - K_kH)(\hat{x_k}^- - x_k) + K_kv_k)((I - K_kH)(\hat{x_k}^- - x_k) + K_kv_k)^T) \\
&= E((I - K_kH)e_k^-(e_k^-)^T(I - K_kH)^T + (I - K_kH)e_k^-(v_k)^TK_k^T + K_kv_k(e_k^-)^T(I - K_kH)^T \\
&+ K_kv_k(v_k)^TK_k^T)
\end{aligned}
$$

因为 $e_k^- = \hat{x_k}^- - x_k = A\hat{x_{k-1}} + Bu_{k-1} - Ax_{k-1} - Bu_{k-1} - w_{k-1}$, 因为 v_k 影响 k 时刻得测量，且 V_k 相互独立，则 e_k 和 v_k 无关从而 $E(e_k v_k) = E(e_k)E(v_k)$ 则

$$
\begin{aligned}
P_k &= E((I - K_kH)e_k^-(e_k^-)^T(I - K_kH)^T + (I - K_kH)e_k^-(v_k)^T K_k^T + K_k v_k (e_k^-)^T (I - K_kH)^T \\
&\quad + K_k v_k (v_k)^T K_k^T) \\
&= E((I - K_kH)e_k^-(e_k^-)^T(I - K_kH)^T) + E((I - K_kH)e_k^-(v_k)^T K_k^T) + E(K_k v_k (e_k^-)^T (I - K_kH)^T \\
&\quad + E(K_k v_k (v_k)^T K_k^T) \\
&= E((I - K_kH)e_k^-(e_k^-)^T(I - K_kH)^T) + K_k R K_k^T
\end{aligned}
$$

我们令 $P_k^- = e_k^-(e_k^-)^T$ 称为先验协方差矩阵, 则可得

$$
\begin{aligned}
P &= (I - K_kH)P_k^-(I - K_kH)^T + K_k R K_k^T \\
&= P_k^- - K_k H P_k^- - P_k^- H^T K_k^T + K_k H P_k^- H^T K_k^T + K_k R K_k^T
\end{aligned}
$$

我们需要方差最小，即 $tr(P)$（代表矩阵得迹）最小

$$
tr(P) = tr(P_k^-) - tr(K_k H P_k^-) - tr(P_k^- H^T K_k^T) + tr(K_k H P_k^- H^T K_k^T) + tr(K_k R K_k^T)
$$

矩阵的转置迹不变则

$$
tr(P) = tr(P_k^-) - 2tr(K_k H P_k^-) + tr(K_k H P_k^- H^T K_k^T) + tr(K_k R K_k^T)
$$

我们进行求偏导通过

$$
\frac{dtr(P_k)}{dK_k} = 0
$$

求解出 K_k 的值,（由于矩阵求导公式 $\frac{dAB}{dA} = B^T$ 和 $\frac{dABA^T}{dA} = 2AB$）

$$
\frac{dtr(P_k)}{dK_k} = 0 - 2tr(HP_k^-)^T + 2tr(K_k H P_k^- H^T) + tr(2K_k R) = 0
$$

于是得到如下公式

$$
\begin{aligned}
K_k R + K_k H P_k^- H^T = H P_k^- \\
K_k = \frac{H P_k^-}{R + H P_k^- H^T}
\end{aligned}
$$

下一个问题是怎么计算先验误差

$$
\begin{aligned}
P_k^- &= E(e_K^-(e_k^-)^T) \\
&= E((A\hat{x_{k-1}} + Bu_{k-1} - Ax_{k-1} - Bu_{k-1} - w_{k-1})(A\hat{x_{k-1}} + Bu_{k-1} - Ax_{k-1} - Bu_{k-1} - w_{k-1})^T) \\
&= E((Ae_{k-1} - w_{k-1})(Ae_{k-1} - w_{k-1})^T) \\
&= E(Ae_{k-1}e_{k-1}^T A^T - w_{k-1}e_{k-1}^T A^T - Ae_{k-1}w_{k-1}^T + w_{k-1}w_{k-1}^T) \\
&= AP_{k-1}A^T + Q
\end{aligned}
$$

其中 w_{k-1} 和 e_{k-1} 是独立的（k 时刻的干扰影响 k+1 时刻的误差）则

$$
P_k^- = AP_{k-1}A^T + Q
$$

因此 **预测：**

$$
\begin{aligned}
\text{先验：} \hat{x_k}^- = A\hat{x_{k-1}} + Bu_{k-1} \\
\text{先验误差协方差：} P_k^- = AP_k^- A^T + Q
\end{aligned}
$$

因为

$$
\begin{aligned}
P &= P_k^- - K_k H P_k^- - P_k^- H^T K_k^T + K_k H P_k^- H^T K_k^T + K_k R K_k^T \\
&= P_k^- - K_k H P_k^- - P_k^- H^T K_k^T + K_k (H P_k^- H^T + R) K_k^T \\
&= P_k^- - K_k H P_k^- - P_k^- H^T K_k^T + P_k^- H^T K_k^T \\
&= P_k^- - K_k H P_k^-
\end{aligned}
$$

校正：

$$\text{卡尔曼增益：} K_k = \frac{P_k^- H^T}{H P_k^- H^T + R}$$

$$\text{后验估计：} \hat{x_k} = \hat{x_k}^- + K_k (Z_k - H \hat{x_k}^-)$$

$$\text{更新误差协方差：} P_k = (I - K_k H) P_k^-$$

1.3 代码内容

Listing 1.1: 使用 Eigen 库实现

```
#include <iostream>
#include <Eigen/Dense>

using namespace Eigen;

class KalmanFilter {
public:
    KalmanFilter(MatrixXd A, MatrixXd B,
        MatrixXd H, MatrixXd Q, MatrixXd R,
        MatrixXd P, VectorXd x)
        : A(A), B(B), H(H), Q(Q), R(R), P(P), x
            (x) {}

    void predict(const VectorXd &u) {
        x = A * x + B * u;
        P = A * P * A.transpose() + Q;
    }

```

```
    void update(const VectorXd &z) {
        VectorXd y = z - H * x; // 计算残差
        MatrixXd S = H * P * H.transpose() + R; // 计算残差协方差
        MatrixXd K = P * H.transpose() * S.inverse(); // 计算卡尔曼增益
        x = x + K * y; // 更新状态估计
        MatrixXd I = MatrixXd::Identity(x.size(), x.size());
        P = (I - K * H) * P; // 更新估计误差协方差
    }

    VectorXd getState() {
        return x;
    }

private:
    MatrixXd A, B, H, Q, R, P;
    VectorXd x;
};

int main() {
    // 定义矩阵
    MatrixXd A(2, 2);
    A << 1, 1,
         0, 1;
```

```
    MatrixXd B(2, 1);
    B << 0.5,
         1;
    MatrixXd H(1, 2);
    H << 1, 0;
    MatrixXd Q(2, 2);
    Q << 1, 0,
         0, 1;
    MatrixXd R(1, 1);
    R << 1;
    MatrixXd P(2, 2);
    P << 1, 0,
         0, 1;
    VectorXd x(2);
    x << 0,
         1;

    // 创建卡尔曼滤波器实例
    KalmanFilter kf(A, B, H, Q, R, P, x);

    // 预测和更新步骤
    VectorXd u(1);
    u << 1;
    VectorXd z(1);
    z << 2;

    kf.predict(u);
```

```
    std::cout << "预测状态：" << kf.getState().transpose() << std::endl;

    kf.update(z);
    std::cout << "更新状态：" << kf.getState().transpose() << std::endl;

    return 0;
}
```

第二章　扩展卡尔曼滤波原理推导

对非线性的模型进行线性化处理，然后进行卡尔曼滤波

2.1　非线性模型

$$
\begin{aligned}
x_k &= f(x_{k-1}, u_k, w_k) \\
z_k &= h(x_k, v_k)
\end{aligned}
$$

其中 u 为输入，z 是输出,w 和 v 是干扰满足如下高斯分布

$$
\begin{aligned}
w &\sim (0, Q) \\
v &\sim (0, R)
\end{aligned}
$$

我们使用泰勒级数来线性化非线性函数即：$f(x) = f(x_0) + \frac{\partial f}{\partial x}|_{x_0}(x - x_0)$, 那在什么地方线性化呢？我们可以想到上一步的后验估计处。

则我们可以得到如下式子（计算时 w_k 视为 0，在（$(\hat{x}_{k-1}, u_k, 0)$）处泰勒展开）

$$
x_k \approx f(\hat{x}_{k-1}, u_k, 0) + \left.\frac{\partial f}{\partial x}\right|_{\hat{x}_{k-1}, u(k), 0}(x_{k-1} - \hat{x}_{k-1}) + w_k
$$

$$
z_k \approx h(\hat{x}_k, 0) + \left.\frac{\partial h}{\partial x}\right|_{\hat{x}_k, 0}(x_k - \hat{x}_k) + v_k
$$

我们令 $A_{k-1} = \left.\frac{\partial f}{\partial x}\right|_{\hat{x}_{k-1},u(k),0}$,$H_k = \left.\frac{\partial h}{\partial x}\right|_{\hat{x}_k,0}$, 则得到了线性化的模型：

$$x_k \approx f(\hat{x}_{k-1}, u_k, 0) + A_{k-1}(x_{k-1} - \hat{x}_{k-1}) + w_k$$

$$z_k \approx h(\hat{x}_k, 0) + H_k(x_k - \hat{x}_k) + v_k$$

2.2　推导

在这个线性化模型上应用卡尔曼滤波算法，我们可以得到如下的扩展卡尔曼滤波算法：

$$\begin{aligned}
\text{预测：} \hat{x}_k^- &= f(\hat{x}_{k-1}, u_k, 0) \\
P_k^- &= A_{k-1}P_{k-1}A_{k-1}^T + Q \\
\text{校正：} K_k &= P_k^- H_k^T (H_k P_k^- H_k^T + R)^{-1} \\
\hat{x}_k &= \hat{x}_k^- + K_k(z_k - h(\hat{x}_k, 0)) \\
P_k &= (I - K_k H_k) P_k^-
\end{aligned}$$

其中 $A_{k-1} = \left.\frac{\partial f}{\partial x}\right|_{\hat{x}_{k-1},u(k),0}$,$H_k = \left.\frac{\partial h}{\partial x}\right|_{\hat{x}_k,0}$ （也需随着每一步更新）