卡尔曼滤波

摘要

本文首先介绍了一个简单实例，用于整体感知卡尔曼滤波器的思想；其次，对卡尔曼滤波的步骤进行详解并绘制了信息流图和系统结构图；最后，模拟直立车角度滤波并给出了结果，从结果可以看出，卡尔曼滤波器可以有效滤除噪声，得到最优估计。

1 小明的故事

这一部分先讲小明的故事。

我是小明，我驾驶一架飞机，初始海拔为 1000 米，以当前海拔为上一次海拔的 0.8 降落，作为驾驶员，我需要直到每个时刻飞机处于什么高度，所以我用下式预测飞机当前时刻的位置：

$$x_k = 0.8\hat{x}_{k-1} \qquad (1.1)$$

但是使用式 (1) 进行预测其实是不准确的，比如风、雨、燃料等因素影响，飞机的实际高度在式 (1.1) 计算结果的基础上总会有一些波动。作为一名训练有素的驾驶员，我要得到飞机位置的准确信息！

所以，我使用了传感器对海拔进行测量，得到当前时刻的海拔的测量值 z_k，测量精度为 10 米，虽然一般来说 10 米的精度已经够用了，但是我希望飞机能够平稳落地，如果在临近落地时，我得到的高度信息稍微偏大了几米，速度还没降下来就落地了，那么可能落地冲击过大，乘客体验不佳，所以传感器提供的信息我也不能完全相信。

直到一位老人给了我一本秘笈，苦心修炼以后，我的飞机每次都是平稳落地，也因此我得到了公司的奖章。现在，我们一起来修炼秘笈吧！

式 (1) 的预测和传感器的测量值是我能获得的两个信息，我想根据这两个信息来得到海拔的准确估计。一种直观的想法就是加权平均：比如当前海拔为式 (1.1) 的预测值的概率为 0.7，为传感器测量值的概率是 0.3，那么我使用式 (1.2) 作为当前海拔的最优估计：

$$\hat{x}_k = 0.7x_k + 0.3z_k \qquad (1.2)$$

那么怎么得到预测值和测量值的概率呢？这就是卡尔曼滤波的核心，即有根据的计算预测值和测量值的概率，使用加权平均得到一个最优估计。

总结：由于噪声干扰，在进行状态预测和测量时，我们得到的信息可能是不准确的，卡尔曼滤波就是结合我们获取的所有信息（状态预测、测量值），给出一个最优估计，使得准确度比单个信息更高。

2 怎么求解

这一部分是卡尔曼滤波的步骤。5 个公式。

假设估计过程噪声和测量噪声相互独立，且服从高斯分布。

整体看来，卡尔曼滤波分为两步，首先是时间更新（即先验估计，包含公式 2.1 和 2.2），然后是量测更新（即后验估计，包含公式 2.3,2.4,2.5）。

a. 状态方程 (用于进行状态预测)

$$\hat{x}_k = A\hat{x}'_{k-1} + Bu_{k-1} \qquad (2.1)$$

其中，A 是上一状态到当前状态的转换矩阵，B 是控制输入到当前状态的转换矩阵，u 变量一般来说是被忽略的；假设状态中每个变量都符合高斯分布，则变量间的相关性可用协方差衡量，状态 $\hat{x}'_{k-1}$ 的协方差矩阵为 P_{k-1}。

b. 预测状态的协方差矩阵

$$P_k = AP_{k-1}A^T + Q \qquad (2.2)$$

其中，P_k 为预测状态的协方差矩阵，即式 (3) 中 $\hat{x}_k$ 的协方差矩阵，Q 为估计过程噪声的协方差矩阵。P_k 表征了由前一状态预测的当前状态的分布。

c. 卡尔曼增益

从上一部分我们知道了卡尔曼滤波会有根据地计算预测值和测量值的概率，测量值的概率为 K，也就是常说的卡尔曼增益，预测值的概率为 $(1-K)$。

首先看一下观测值及其分布：

$$y = H\hat{x}_k$$
$$\sum\nolimits_0 = H_k P_k H_k^T$$

然后，测量噪声的协方差为 R，则可得到卡尔曼增益 K 为：

$$K = \sum\nolimits_0 \cdot (\sum\nolimits_0 + R)^{-1}$$

结合后三个公式可对 K 进行精简，为了后面的公式表示方便，这里使用精简后的表达式：

$$K' = P_K H^T (HP_k H^T + R)^{-1} \qquad (2.3)$$

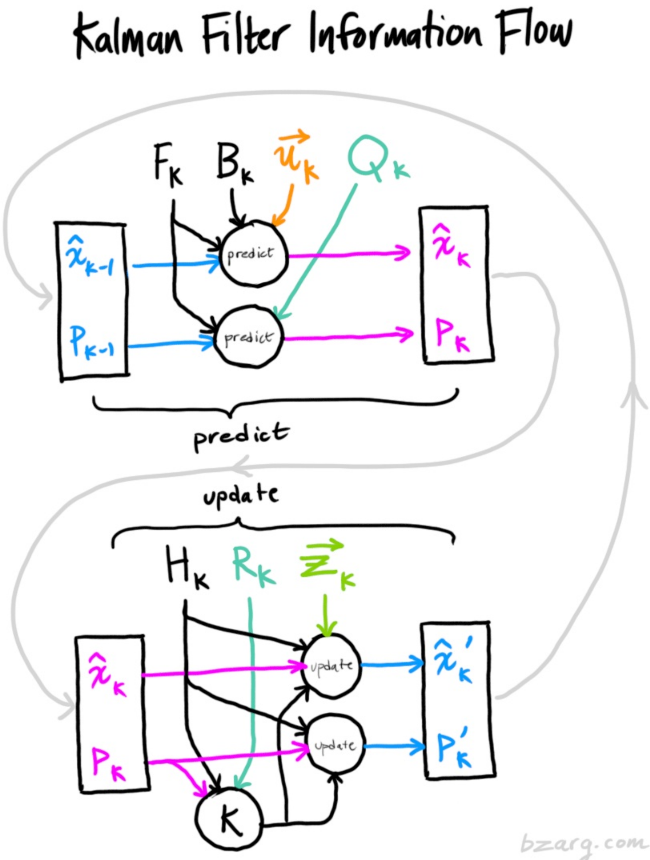
d. 最优估计值 y'

$$\hat{x}'_k = \hat{x}_k + K'(z_k - H\hat{x}_k) \qquad (2.4)$$

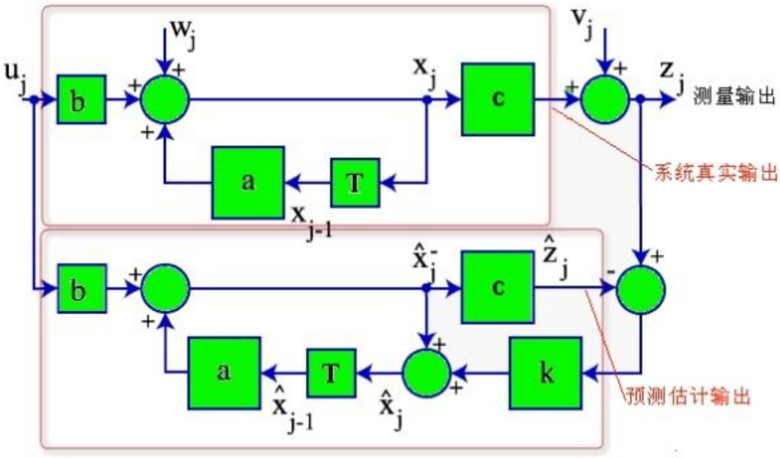
e. 协方差矩阵更新

$$P'_k = P_k - K' H_k P_k \qquad (2.5)$$

总结：上述 5 个公式在卡尔曼滤波的信息流图完美体现 (注：图中的 $F_k \quad A_k$，本文未对卡尔曼增益进行推导，感兴趣可查看参考部分“卡尔曼滤波详解”链接)。



3 卡尔曼滤波系统结构图



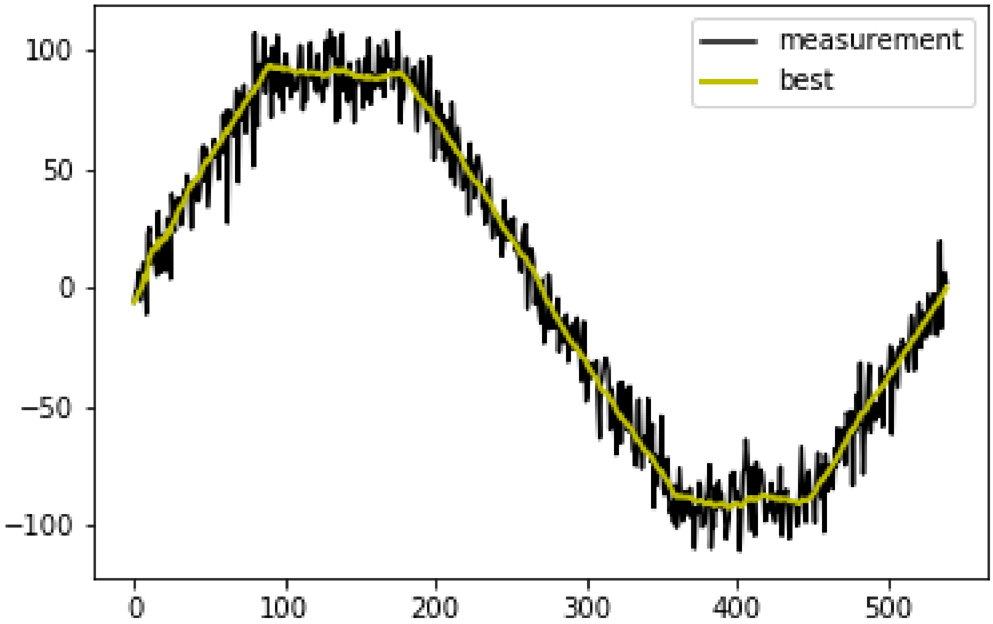
4 示例：模拟直立车角度滤波

a.Python 程序

```
# 定义函数
    def Kalman6(Accel,Gyro,Q,R,dt):
    # Accel:传感器测得的角度
    # Gyro: 传感器测得的角速度
    # Q:(2,2)的对角阵
    # R:Scalar
        A=np.array([[1,-dt],[0,1]])    #(2,2)
        B=np.array([[dt],[0]])    #(2,1)
        H=np.array([[1,0]])    #(1,2)
        I=np.array([[1,0],[0,1]])

        Xp=np.array([[s[0]],[0]])    #(2,1)
        Pp=np.array([[1,0],[0,1]])
        T=np.zeros((len(s)))
        T[0]=s[0]

        for i in range(1,len(s)):
            Xu=A.dot(Xp)+B*Gyro[i]
            Pu=A.dot(Pp).dot(A.T)+Q

            K=Pu.dot(H.T)*1/(H.dot(Pu).dot(H.T)+R)

            Xp=(I-K.dot(H)).dot(Xu)+K.dot(s[i])
            Pp=(I-K.dot(H)).dot(Pu)

            T[i]=Xp[0]
        return T

    # 生成数据（时间步长为1）
```

```
# 生成角速度数据
x1=np.ones(90)
x2=np.zeros(90)
x3=-np.ones(180)
x4=np.zeros(90)
x5=np.ones(90)
Gyro=np.concatenate([x1,x2,x3,x4,x5])
# 生成角度测量值数据（模拟加速度传感器测量角度）
s1=np.linspace(1,90,90)
s2=90*np.ones(90)
s3=np.linspace(90,-89,180)
s4=-89*np.ones(90)
s5=np.linspace(-89,0,90)
angle=np.concatenate([s1,s2,s3,s4,s5])
noise=np.random.normal(0,10,540)
s=angle+noise
Accel=s
# 设置角度噪声和陀螺仪漂移噪声的协方差
Q_angle=0.0002
Q_gyro=0.0001
Q=np.array([[Q_angle,0],[0,Q_gyro]])
# 设置测量噪声的协方差
R=100

# 设置时间步长
dt=1
# 利用卡尔曼滤波得到角度的最优估计
y=Kalman6(Accel,Gyro,Q,R,dt)
plt.plot(range(540),s,c='k',label='measurement')
plt.plot(range(540),y,c='best',linewidth=2)
plt.legend()
```

b. 结果 (动图可查看”result2.gif”)



5 参考

[图说卡尔曼滤波]https://www.zhihu.com/collection/428837658

[卡尔曼滤波详解]https://blog.csdn.net/honyniu/article/details/88697520

[角度滤波 C 代码分析]https://wenku.baidu.com/view/e960c85959fafab069dc5022aaea998fcc2240e4.html