Kalman Filter 基础知识

1 卡尔曼滤波简介

本小节[1]先简要介绍卡尔曼滤波（Kalman Filtering，简称：KF），下一小节将介绍 EKF 并将其用于参数的学习。关于 KF 和 EKF 的详细描述可参阅文献: [1]SIMON D. Optimal state estimation: Kalman, H [infinity] and nonlinear approaches[M]. Hoboken, N.J: Wiley-Interscience, 2006.

KF 可以由递归最小二乘（RLS）简单扩展得到。在 LS 中，测量的增多会导致极大的计算量，而 RLS 则可以递归地对状态进行估计。即，对于一个状态恒为 x 的系统，其测量方程为：

$$y_k = H_k x + v_k \tag{1}$$

式中，y_k 为 k 时刻的测量值，H_k 为 k 时刻的测量矩阵，v_k 为测量误差，其协方差矩阵记为 R_k。RLS 的目标是寻找 k 时刻的增益矩阵 K_k，使得 k 时刻的状态估计 $\hat{x}_k$ 与 $k-1$ 时刻的状态估计 $\hat{x}_{k-1}$ 满足：

$$\hat{x}_k = \hat{x}_{k-1} + K_k(y_k - H_k\hat{x}_{k-1}) \tag{2}$$

若以 $\epsilon_{x,k} = x - \hat{x}_k$ 代表 k 时刻的估计误差，$P_k = E(\epsilon_{x,k}\epsilon_{x,k}^T)$ 代表此刻的估计误差协方差矩阵，则最小化目标函数 $Tr(P_k)$[2]可得[3]：

$$K_k = P_{k-1}H_k^T(H_kP_{k-1}H^T + R_k)^{-1} \tag{3}$$

$$P_k = (I - K_kH_k)P_{k-1}(I - K_kH_k)^T + K_kR_kK_k^T \tag{4}$$

若放松 RLS 中状态 x 恒定的假设，令：

$$x_k = F_{k-1}x_{k-1} + G_{k-1}u_{k-1} + \omega_{k-1} \tag{5}$$

[1] 为了与常规用法一致，本节使用 x 代表状态。

[2] $E[(x_1-\hat{x}_1)^2+\ldots+(x_n-\hat{x}_n)^2] = E[\epsilon_{x_1,k}^2+\ldots+\epsilon_{x_n,k}^2] = E[\epsilon_{x,k}^T\epsilon_{x,k}] = E[Tr(\epsilon_{x,k}\epsilon_{x,k}^T)] = Tr(P_k)$.

[3] 此处亦可理解为 RLS 的测量更新方程。

式中，u_k 为已知输入，ω_k 为协方差为 Q_k 的零均值高斯白噪声。此时，(1) 亦变为：

$$y_k = H_k x_k + v_k \tag{6}$$

那么，状态 x_k 的均值的更新方程及其协方差的时间更新方程为：

$$\bar{x}_k = F_{k-1}\bar{x}_{k-1} + G_{k-1}u_{k-1} \tag{7}$$

$$P_k = F_{k-1}P_{k-1}F_{k-1}^T + Q_{k-1} \tag{8}$$

在卡尔曼滤波（KF）的推导中，为了突出时间更新和测量更新，对测量前后状态相关的值加以区分：$\hat{x}_k^-$ 表示 k 时刻的测量值到来之前对状态 x_k 的估计值，$\hat{x}_k^+$ 表示考虑 k 时刻的测量值 y_k 后对状态 x_k 的重新估计值。相应地，P_k^- 表示 $\hat{x}_k^-$ 的估计误差的协方差，P_k^+ 表示 $\hat{x}_k^+$ 表示 $\hat{x}_k^+$ 的估计误差的协方差，即：

$$P_k^- = E[(x_k - \hat{x}_k^-)(x_k - \hat{x}_k^-)^T] \tag{9}$$

$$P_k^+ = E[(x_k - \hat{x}_k^+)(x_k - \hat{x}_k^+)^T] \tag{10}$$

而 KF 的递推公式只需要在 RLS 的基础上稍加更改并结合 (7) 和 (8) 得到。KF 的时间更新方程为：

$$\hat{x}_k^- = F_{k-1}\hat{x}_{k-1}^+ + G_{k-1}u_{k-1} \tag{11}$$

$$P_k^- = F_{k-1}P_{k-1}^+F_{k-1}^T + Q_{k-1} \tag{12}$$

其测量更新方程为：

$$K_k = P_k^- H_k^T (H_k P_k^- H^T + R_k)^{-1} \tag{13}$$

$$P_k^+ = (I - K_k H_k) P_k^- (I - K_k H_k)^T + K_k R_k K_k^T \tag{14}$$

$$\hat{x}_k^+ = \hat{x}_k^- + K_k(y_k - H_k \hat{x}_k^-) \tag{15}$$

当 $k = 1$ 时，KF 需要设置初始化参数：

$$\hat{x}_0^+ = E(x_0) \tag{16}$$

$$P_0^+ = E[(x_0 - \hat{x}_0^+)(x_0 - \hat{x}_0^+)^T] \tag{17}$$

当 $k = 1, 2, 3, \ldots$ 时，只需要迭代上述时间更新方程和测量更新方程即可获得 k 时刻的状态估计值 $\hat{x}_k^+$。

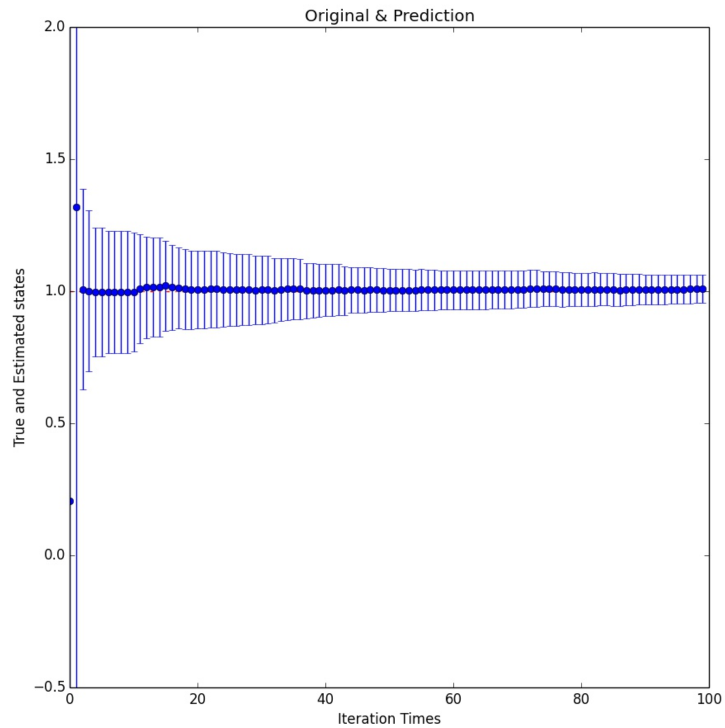

图 1: 状态 a 的估计。

2 Kalman for curve fitting（抛物线拟合）

我们认为其 parameters，即 states 为 (a, b, c) 恒定（实际上把 Kalman 当 RLS 来用啦），而测量方程为：

$$y[k] = ax[k]^2 + bx[k] + c + w[k] \tag{18}$$

每一次的测量相当于 $y[k]$. 在 code 中，我们假设 $(a, b, c) = (1, 2, 3)$, 得到 100 组测量数据后，最终的 estimate 为 $[1.00948328, 1.95366774, 3.04191686]$，每一次 iteration 的结果和 error bar 图 1 所示[4]。

[4]仅 plot 了第一个状态即 $a = 1$ 的 estimation 结果，其它两个类似就不画。

3 附录代码

注意第 33 行只是简单地将协方差矩阵进行传播，并没有使用时间更新方程。

```
1  # coding=utf-8
2  import matplotlib.pyplot as plt
3  import numpy as np
4  def kalman(A, y):
5      """
6      Formulars from the book《Optimal State Estimation》
7      Page88 & P128
8      :param A: array in the sklearn form. each "row"
9       represents an input (the measurement coeffcient).
10     :param y:vector of the response variable (measuremen)
11     :return: x:the estimates. p: covariance of x
12     """
13     dim = A.shape[1]
14     n = A.shape[0]
15     # initialize the estimate of the state(s)
16     # and its(their) covirance
17     x0 = np.zeros((dim, 1))
18     P0 = 1e5 * np.identity(dim)
19     R = 1  # the measurement variance
20     # convert numpy arrays to matrix(suitable for matrix manupulation)
21     x0 = np.mat(x0)
22     P0 = np.mat(P0)
23     # start iteration
24     x, p = [], []
25     for i in range(n):
26         Hk = np.mat(A[i, :][:, np.newaxis]).T
27         Kk = P0 * Hk.T * (Hk * P0 * Hk.T + R).I
28         x_new = x0 + Kk * (y[i] - Hk * x0)
29         Pk = (np.mat(np.identity(dim)) - Kk * Hk) * P0 * \
```

```
            (np.mat(np.identity(dim)) - Kk * Hk).T + Kk * R * Kk.T
        #Note! this is not universal! We assume that the "time update"
        #step is trival, i.e., the states are constants
        #Modify the following line as you need.
        P0 = Pk
        x0 = x_new
        # for return
        x.append(x0)
        p.append(P0)
    #convert x back to array
    x=[np.array(xi).squeeze() for xi in x]
    return x,p
def testKalman():
    #prepare training data (3-dimension)
    dataLen = 100
    A = np.random.random_sample((dataLen, 1)) * 5
    A = np.c_[A ** 2, A, np.ones((dataLen, 1))]
    x = np.array([1, 2, 3])
    y = np.dot(A, x) + np.random.randn(dataLen) * 0.1
    x_,p= kalman(A, y)
    print "final estimate:",x_[-1]
    #extract the confidence interval of the 0th state
    y_err=[np.sqrt(pi[0,0]) for pi in p]
    #now plot!
    fig, ax = plt.subplots(1,1,figsize=(10,10))
    ax.plot(range(dataLen),np.ones(dataLen)*x[0], 'r-.')
    # ax.plot(range(dataLen),[xi[0] for xi in x_], 'b-x', alpha=0.5)
    ax.errorbar(range(dataLen),[xi[0] for xi in x_], yerr=y_err, fmt='o')
    ax.set_ylim([-0.5,2])
    ax.set_title('Original & Prediction')
    ax.set_xlabel(u'Iteration Times')
    ax.set_ylabel(u'True and Estimated states')
    plt.show()
```

```
    pass
if __name__ == '__main__':
    testKalman()
```