Kalman Filter 实战作业

I. 引言

庆丰三年夏，董老师小讲 kalman，谓吾等自行操练 Kalman 的 formula，故有此记。

II. 初识 Kalman

我研一已经看过了有关 Kalman 的 books，只记得其时间更新以及状态更新的思想很深刻，过去两年后又有了新的感悟。在《Optimal State Estimation》一书中，作者是从递归最小二乘 (RLS)，很自然地过渡到 Kalman Filter 的。RLS 产生的一个重要需求就是要递归估计，而不是 batch mode，即：

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

问题转化为如何求K_k.

当然了，对于一个 estimation 任务，很自然的要先确定如何衡量这个 estimation 的好坏，而这个衡量标准（criterion）又可以作为我们的目标函数。在 RLS 中，如果将估计误差（即$x-\hat{x}$) 的期望$E(\epsilon_{x,k})$ 作为目标函数，对于零均值的测量噪声以及理想的状态初始值来说，则在对$E(\epsilon_{x,k})$ 进行递归估计时得不出什么重要信息（详见 P84). 因此不得不另选一个目标函数，即估计误差的方差之和$J_k = Trace(P_k)$。通过简单的求导即可得到更新公式：

$$K_k = P_{k-1}H_k^T(H_kP_{k-1}H^T + R_k)^{-} \tag{3}$$
$$\hat{x}_k = \hat{x}_{k-1} + K_k(y_k - H_k\hat{x}_{k-1}) \tag{4}$$
$$P_k = (I - K_kH_k)P_{k-1}(I - K_kH_k)^T + K_kR_kK_k^T \tag{5}$$

当然，在 RLS 中，我们假定要估计的 state 是 constant 的。what if not? Then comes the Kalman Filter! 也就是说 RLS 中我们只有测量方程，现在需要考虑加入时间更新方程：

$$x_k = F_{k-1}x_{k-1} + G_{k-1}u_{k-1} + \omega_{k-1} \tag{6}$$

此时更新公式只需要加入:（详见 P128)

$$P_{k-1} = F_{k-1}P_{k-1}^1F_{k-1}^T + Q_{k-1} \tag{7}$$

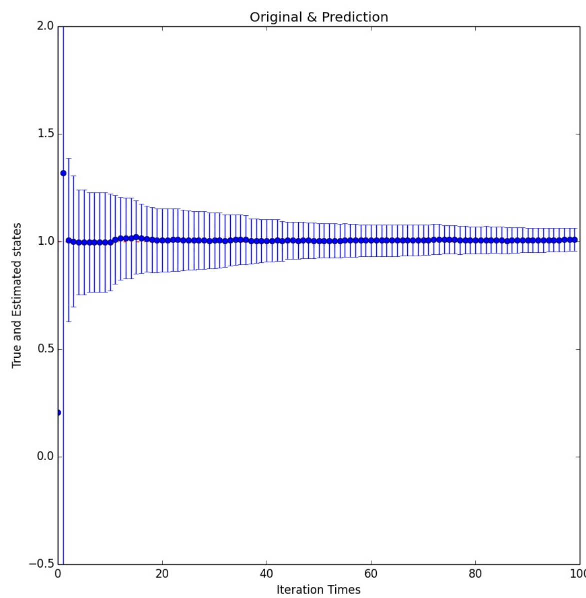
其中P_{k-1}^1 即上一次测量更新得到的P_{k-1}，这里为了和 RLS 的 notation 一致，加入了上标 1. 现在回顾一下，RLS 和 Kalman 的区别就是，在 RLS 中，直接令$P_{k-1} = P_{k-1}^1$（因为状态是恒定的嘛），在 Kalman 中加入了时间更新！这个更新是基于我们对这个 system 的 dynamics 的了解来的！

III. 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：

$$y[k] = ax[k]^2 + bx[k] + c + w[k] \tag{8}$$

每一次的测量相当于$[x[k]^2, x[k], 1 \mid y[k]]$. 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 如下图所示（注，仅 plot 了第一个状态即$a = 1$ 的 estimation 结果，其它两个类似就不画了）。



IV. 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
```

```
    # convert numpy arrays to matrix(suitable for matrix manupulation)
    x0 = np.mat(x0)
    P0 = np.mat(P0)
    # start iteration
    x, p = [], []
    for i in range(n):
        Hk = np.mat(A[i, :][:, np.newaxis]).T
        Kk = P0 * Hk.T * (Hk * P0 * Hk.T + R).I
        x_new = x0 + Kk * (y[i] - Hk * x0)
        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()
```