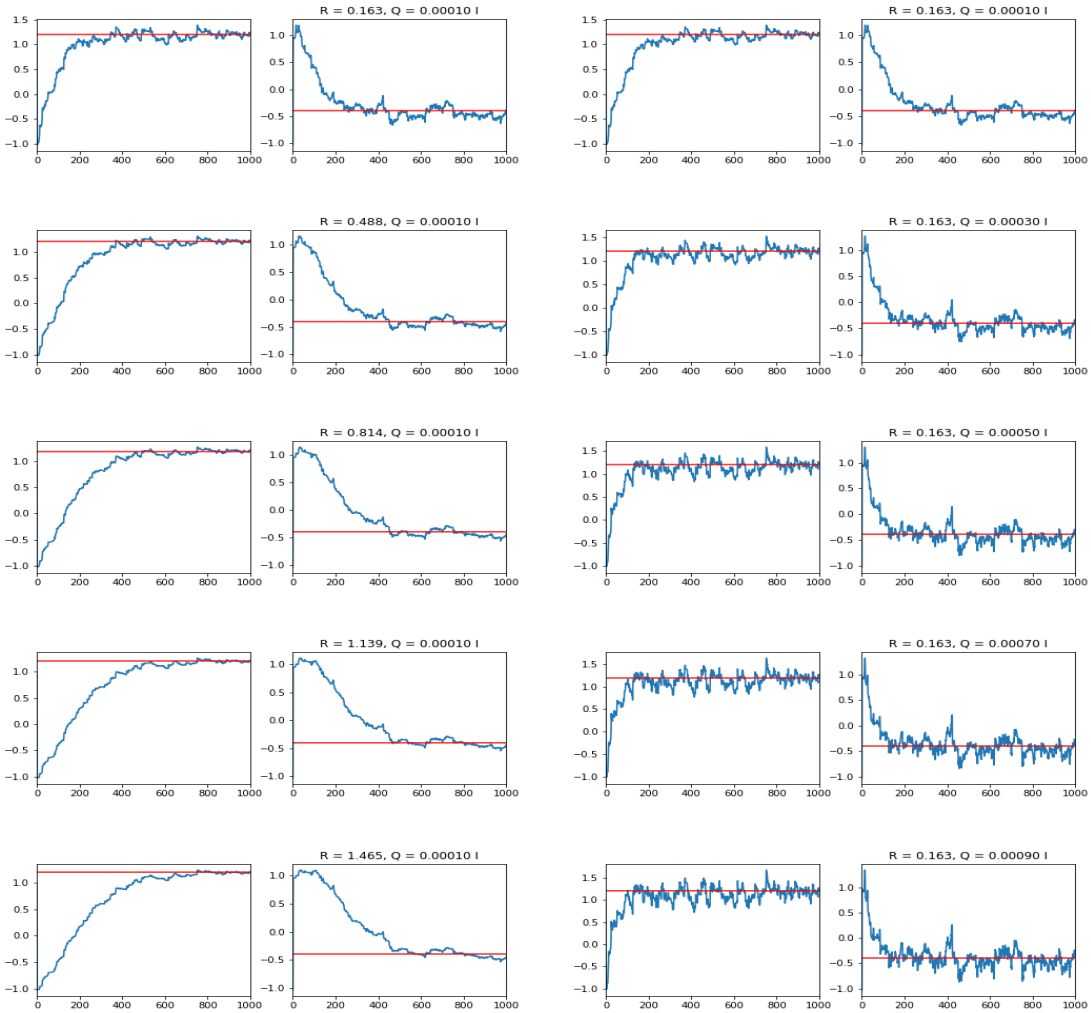
Lab Report–Kalman and Particle Filters; Online PCA

Zhou Jie 31384579 jz5n19@soton.ac.uk

1 Kalman Filter

In order to observe how the parameter convergence, 10 groups of different process noise covariance($\mathbf{Q}$) and measurement noise(R) were used when I implemented Kalman filter.

Left column: different R ,same $\boldsymbol{Q}$ *Right column: same R ,different* $\boldsymbol{Q}$



The left column of the figure above showed the converge speed under different R and the same **Q**. The value of R is tuned by changing the fraction of data variance (first 10 samples), the original fraction is 0.2, i.e.,

```
R = 0.2*np.std(ex[0:10])
```

And the corresponding values of R for different fraction are showed below:

fraction	0.2	0.6	1.0	1.4	1.8
R	0.163	0.488	0.814	1.139	1.465

We can see, as the R increase, it takes more data for the parameters to converge, i.e., the converge speed became slower.
The right column showed the effect of different **Q** while R remains unchanged. The **Q** was tuned by changing the value of beta in the program, this is the the original beta and **Q**:

```
beta = 0.0001
Q = beta*np.eye(2)
```

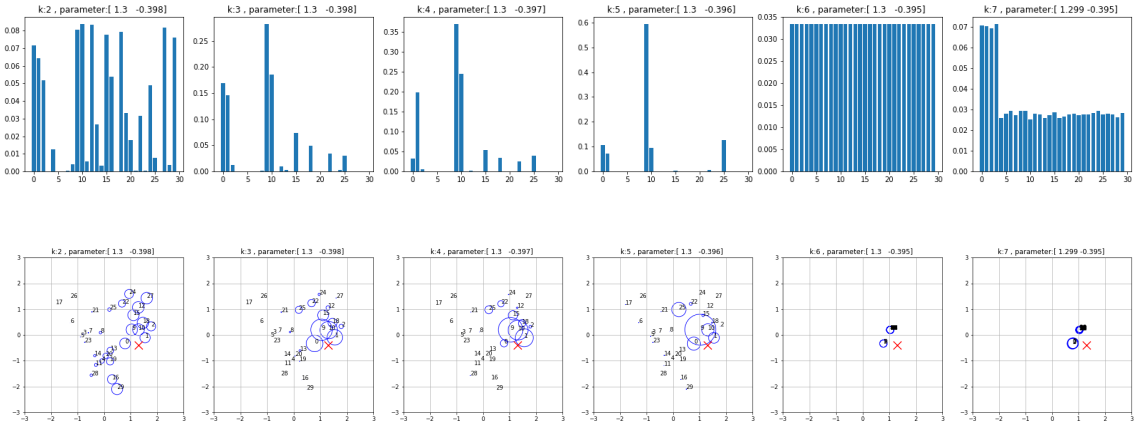
and 5 different betas were used to control the value of Q:

beta	0.0001	0.0003	0.0005	0.0007	0.0009

We can see, as the beta increased, the fluctuation of the curve also increased.

2 Particle Filter

The graphs below showed a obvious weights degeneracy from the 2nd to the 5th samplings. Almost all the weights are concentrated on a few of the 30 particles in the 5th sampling. After implementing resampling in the 6th sampling, the degeneracy had been improved:



The corresponding positions of particles are showed on the 2nd row of the graph: the weights are expressed by the size of circle, the indices of particles are denoted, and the true value of

parameter are marked as a red cross on the scatter plot.
We can see the weights of particles around the parameter become larger as the sequential sampling progresses. Upon the 6th sampling, the effective sample size N_{eff} is less than 2(the threshold used in my code), and the resampling happened, the particles with large weights in the 5th sampling became the parents of the particles in the following sampling. That's why all the particles are concentrated together in the 6th and 7th samplings.

3 Extended Kalman Filter

The pseudocode of the extended Kalman filter algorithm for the logistic regression problem is described below:

Algorithm 1 Extended Kalman filter estimates of logistic function parameters

$Ys \leftarrow \begin{bmatrix} x_{11} & x_{12} \\ \dots & \dots \\ x_{N1} & x_{N2} \end{bmatrix}$ ▷ the input of the observation equation(synthetic data)

$p\_Ys \leftarrow \begin{bmatrix} y_{11} & y_{12} \\ \dots & \dots \\ y_{N1} & y_{N2} \end{bmatrix}$ ▷ the output of the observation equation(synthetic data)

$y \leftarrow p\_Ys$

$x \leftarrow \begin{bmatrix} 0 \\ 0 \end{bmatrix}$

$th\_n1\_n1 \leftarrow \begin{bmatrix} randn() \\ randn() \end{bmatrix}$ ▷ return a 2×1 vector from standard normal distribution

$P\_n1\_n1 \leftarrow 0.001 * \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

$R \leftarrow 0.2 * \text{std}(\text{noise}[0{:}10])$ ▷ guess the variance from observation noises

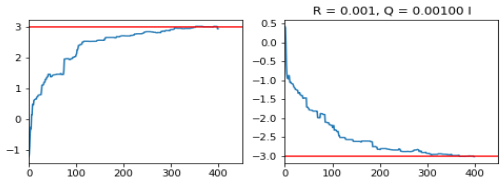
$Q \leftarrow 0.001 * \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

$th\_conv \leftarrow \begin{bmatrix} th\_n1\_n1[0] & th\_n1\_n1[1] & 0 & \dots & 0 \\ th\_n1\_n1[0] & th\_n1\_n1[1] & 0 & \dots & 0 \end{bmatrix}$ ▷ a 2×N matrix for storing values

for `n=0:N` **do**

- $x[0], x[1] \leftarrow Ys[n, 0], Ys[n, 1]$
- $th\_n\_n1 \leftarrow th\_n1\_n1$
- $P\_n\_n1 \leftarrow P\_n1\_n1 + Q$
- $yh \leftarrow \text{sigmoid}(th\_n\_n1, x)$
- $en \leftarrow y[n] - yh$
- $H \leftarrow (\text{sigmoid}(th\_n\_n1, x) * (1 - \text{sigmoid}(th\_n\_n1, x))) * x.\text{T}$ ▷ the derivative of sigmoid
- $den \leftarrow H@P\_n\_n1@H.\text{T} + R$
- $kn \leftarrow P\_n\_n1@H.\text{T} / den$
- $th\_n\_n \leftarrow th\_n\_n1 + kn * en$
- $P\_n\_n \leftarrow (\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} - kn@H)@P\_n\_n1$
- $th\_conv[0, n] \leftarrow th\_n\_n[0]$
- $th\_conv[1, n] \leftarrow th\_n\_n[1]$
- $th\_n1\_n1 \leftarrow th\_n\_n$
- $P\_n1\_n1 \leftarrow P\_n\_n$

By implementing the EKF on the synthetic data with α=1.5 and 400 datapoints, I got the converge curve of the parameters estimation like below:



4 Online PCA

I chose the 'semi-conductor' data(http://archive.ics.uci.edu/ml/datasets/SECOM) from UCI Repository of Machine Learning Datasets for studying online PCA. Every example in this dataset has 591 features. As a preprocessing, the NA values had been filled by the mean of the column.

Parameters required by the algorithm

In addition to the dataframe, the input parameters of algorithm 1 in the paper also include the square of the frobenius norm of the dataframe: $\|\mathbf{X}\|_F^2 = 4.91 \times 10^{11}$, the maximum square of the norm of the features:$\|\mathbf{x}_t\|_2^2 = 1.22 \times 10^{11}$, and the target dimension l.

The low dimensional representations

In order to figure out the value of l, the upper and lower bounds need to be specified. According to the assumption of algorithm, the target low dimension $l \leq \|\mathbf{X}\|_F^2 / \|\mathbf{x}_t\|_2^2 = 4.02$. So the upper bound is clear. For the lower bound, by implementing offline PCA with sklearn,if 92% of the variance is retained, the target low dimension will be 3, i.e., $k = 3 < l \leq 4.02$. So a proper value for l is 4.

Computational savings

The computational savings of online PCA is mainly reflected on the space complexity. The offline PCA requires $\Omega(d^2)$ auxiliary space (in memory) , where d is the number of features in the dataframe, but the online PCA requires only $\mathcal{O}(d^2)$ auxiliary space.