COMP6247 Kalman and Particle Filters; Online PCA

DAJI LI
Student ID: 31326234
E-mail: dl5n19@soton.ac.uk

March 2020

1 Kalman filter

I generated 2 second order autoregressive process and implement Kalman filter for them, we can find that the will parameter $\boldsymbol{\theta}$ will converge according to the true value a, no matter a is a constant or time varying which shown in Fig. 1.

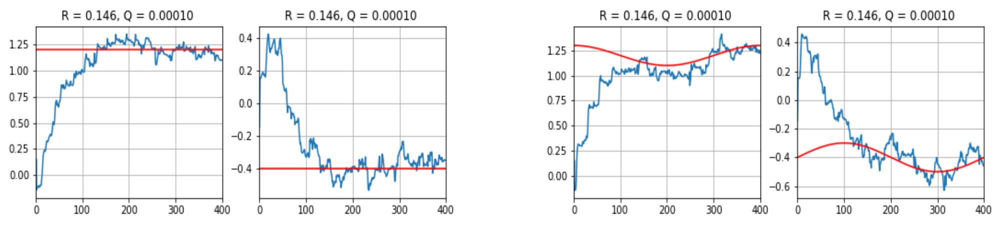

Figure 1: Kalman filter with different second order Autoregressive Process: the left is Autoregressive Process and the right is time varying Autoregressive Process.

$$P(n|n-1) = P(n-1|n-1) + Q \tag{1}$$

$$\boldsymbol{k}(n) = \frac{P(n|n-1)\boldsymbol{x}(n)}{R + \boldsymbol{x}(n)^T P(n|n-1)\boldsymbol{x}(n)} \tag{2}$$

In Fig. 2, I implement initial status with 5 different process noise covariance R and measurement noise variance Q and same initial parameter $\boldsymbol{\theta}$. We can find if we only modify process noise covariance R, the convergence speeds are inversely related to process noise covariance R. And the same time, we can find if we only modify measurement noise variance Q, the convergence speeds are proportional to noise variance Q. In fact, we can refer to equation (1) and (2), we can find the convergence speeds are proportional to the process noise covariance dividing by measurement noise variance R/Q, the convergence speeds are inversely related to R/Q. In Fig. 2, **the values of R/Q in each row are same, we can find the convergence speeds decrease with the values of R/Q increasing but the fluctuations are also stronger. Vice versa.**

2 Particle filter

Particle filters (or Sequential Monte Carlo) are a family of online Bayesian inference tools in which a non-parametric density represented by samples drawn from it is propagated over time along with importance weights. In this practice, we use Sequential Importance Sampling (SIS) Algorithm, we can implement the degeneracy of particle weight via condensation algorithm which the weights show in Fig. 3. The key idea is to represent the required posterior density function by a set of random samples with associated weights and to compute estimates based on these samples

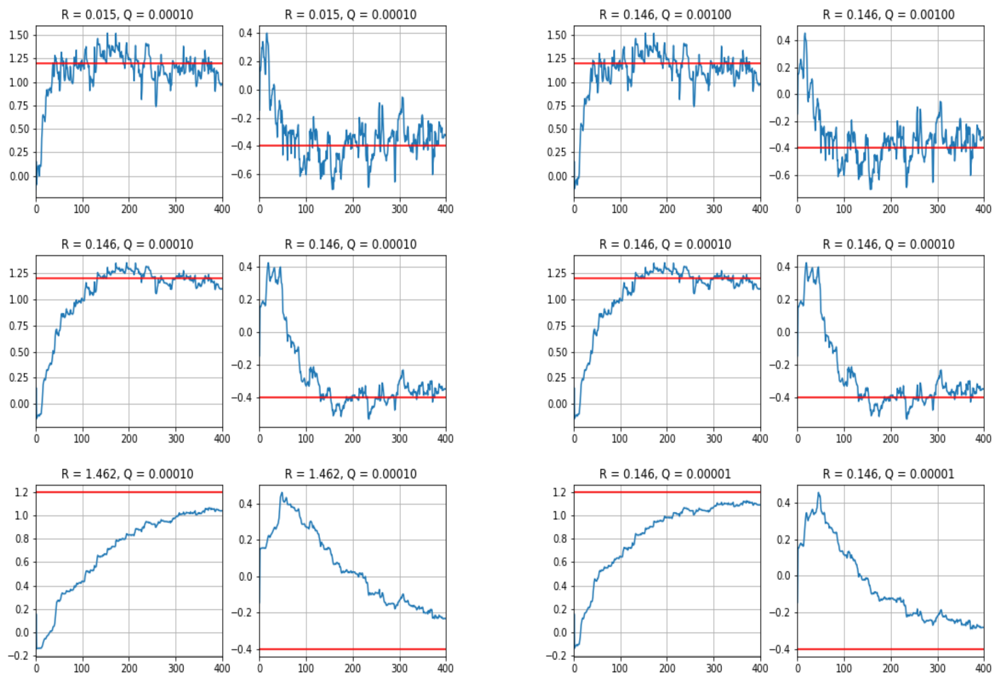

Figure 2: Kalman filter with different initial values of the process noise covariance and measurement noise variance: the lefts are about process noise covariance R and the rights are about measurement noise variance Q. (same value of R/Q for each row)

and weights. In this report, we use the likelihoods function $w_k^i = w_{k-1}^i p\left(\mathbf{z}_k|\mathbf{x}_k^i\right)$. And then we can estimate dynamic model of the second order AR process by **the positions of particles dot associated weights**. In fact, the number of samples is relative with accuracy of filter but a bigger number of samples will increase time complexity.

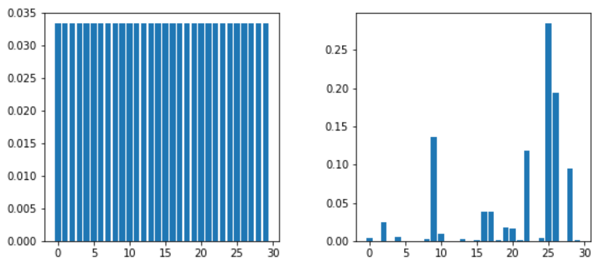

Figure 3: The weights of initial particle and degeneracy: the left is initial particle and the right is degeneracy of particle weight. (iteration=400)

However, a common problem with the SIS particle filter is the degeneracy phenomenon, where after a few iterations, all but one particle will have negligible weight. It has been shown that the variance of the importance weights can only increase over time, and thus, it is impossible to avoid the degeneracy phenomenon. And the choice of importance density is also important. So we use algorithm 2 (Resampling Algorithm) to avoid the degeneracy of weight in excess, which shows in Fig. 4. And before resampling, we also need to judge the degree of degeneracy by $\widehat{N_{eff}} = \frac{1}{\sum_{i=1}^{N_s}\left(w_k^i\right)^2}$ and if $\widehat{N_{eff}} < N_T$, we implement resampling. In my program, I set $N_T = 3$. In the Fig. 4, we can see the final particles position.

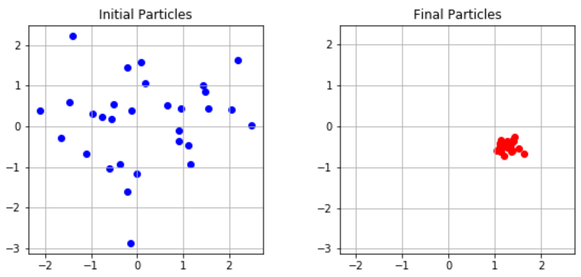

Figure 4: The distribution of initial particle and resampling particle. (iteration=400)

In Fig. 5, we use implement particle filter to normal AR process (left) and time varying AR process signal (right), which we can find the prediction a following the AR signal.

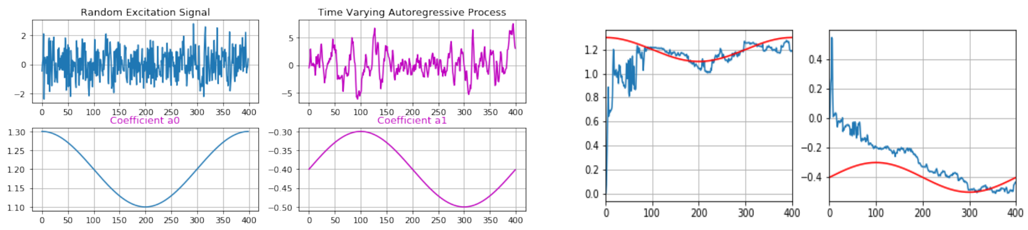

Figure 5: Particle filter with time varying second order Autoregressive Process ($N_T = 3$).

3 Extended Kalman filter

I generate data from a two Gaussian class problem in two dimensions with means set at $\begin{bmatrix} -\alpha & \alpha \end{bmatrix}$ and $\begin{bmatrix} \alpha & -\alpha \end{bmatrix}$ with common covariance marix $\begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}$, which the distribution shows in Fig.5 (left) with $\alpha = 1$ and the blue points are class 1 and the pink points are class 0. And Fig. 6 (right) shows the distribution of logistic model by extended Kalman filter with a random walk state dynamics and Gaussian noise processes.

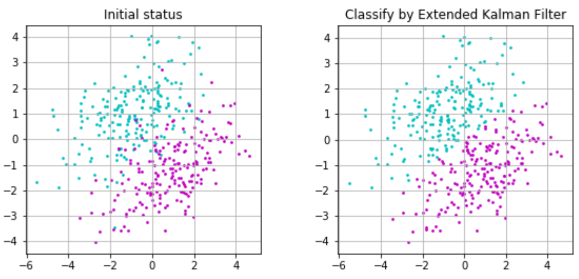

Figure 6: The distribution of two Gaussian class: the left is the initial status and the right is the classification by extended kalman filter.

I implement the extended Kalman filter shown in Alg. 1, which is different with original Kalman filter in calculating the Jacobian matrix $\hat{H}_k$ according to equation (3) and (4), and using logistic model to calculate the prediction.

$$\mathbf{h}_k(x) = \frac{1}{1 + \exp\left(-\theta^T x\right)} \tag{3}$$

$$\hat{H}_k = \left.\frac{d\mathbf{h}_k(x)}{dx}\right|_{x=\boldsymbol{\theta}_{k|k-1}} = \left(\frac{1}{1+e^{-\boldsymbol{\theta}^T\boldsymbol{x}}}\right)' = \frac{1}{1+e^{-\boldsymbol{\theta}^T\boldsymbol{x}}}\cdot\frac{e^{-\boldsymbol{\theta}^T\boldsymbol{x}}}{1+e^{-\boldsymbol{\theta}^T\boldsymbol{x}}}\cdot\boldsymbol{x} = \mathbf{h}_k(x)(1-\mathbf{h}_k(x))\boldsymbol{x} \tag{4}$$

Algorithm 1 Extended Kalman filter for logistic regressive classification in each iteration.

```
Input: $\boldsymbol{\theta}_{k-1|k-1}, P_{k-1|k-1}, \boldsymbol{\theta}_{k|k-1}, z_t$
Output: $\boldsymbol{\theta}_{k|k}, P_{k|k}$;
1: Predict the parameter $\boldsymbol{\theta}_{k|k-1}$ and the covariance matrix $P_{k|k-1}$: $\boldsymbol{\theta}_{k|k-1} = \boldsymbol{\theta}_{k-1|k-1}$ and $P_{k|k-1} = P_{k-1|k-1} + Q_{k-1}$;
2: Calculate the Jacobian Matrix $\hat{H}_k$: $\hat{H}_k = \left.\frac{d\mathbf{h}_k(x)}{dx}\right|_{x=\boldsymbol{\theta}_{k|k-1}} = \mathbf{h}_k(x)\left(1-\mathbf{h}_k(x)\right)\boldsymbol{x}$ and the denominator of Kalman gain $S_k$: $S_k = \hat{H}_k P_{k|k-1}\hat{H}_k^T + R_k$;
3: Calculate the Kalman gain $K_k$: $K_k = P_{k|k-1}\hat{H}_k^T S_k^{-1}$;
4: Update $\boldsymbol{\theta}_{k|k} = \boldsymbol{\theta}_{k|k-1} + K_k\left(\mathbf{z}_k - \mathbf{h}_k(x)\right)$ and $P_{k|k} = P_{k|k-1} - K_k\hat{H}_k P_{k|k-1}$;
       return $\boldsymbol{\theta}_{k|k}, P_{k|k}$;
```

In Fig. 7 (left) is the result of cross entropy loss function, which shows the total errors to true values. If we plot the logistic model with class in 3 dimension, we can find the class boundary showed in Fig. 7 (right).

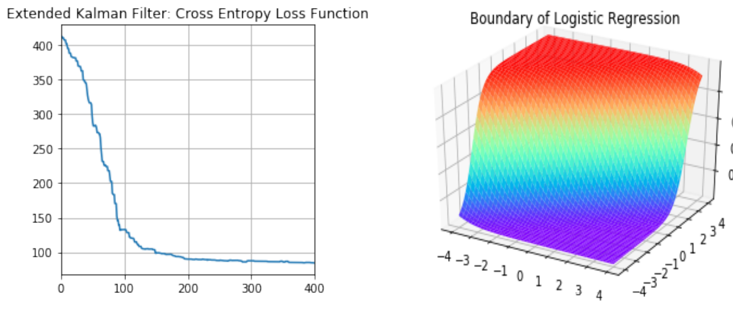

Figure 7: Logistic regression: the left is the cross entropy loss function and the right is the result of logistic model in 3 dimension.

4 Online PCA

In online PCA (OPCA) of this paper the setting is identical except for two differences with original PCA: i) the vectors $\mathbf{x}_t$ are presented to the algorithm one by one and for every presented $\mathbf{x}_t$ the algorithm must output a vector $\mathbf{y}_t$ before receiving $\mathbf{x}_{t+1}$; ii) the output vectors $\mathbf{y}_t$ are l dimensional with $\ell \geq k$ to compensate for the handicap of operating online. By using this way, we can save large space complexity, we cannot input the whole data in memory if the data set is huge. For time complexity, To train large data set in offline PCA, the bottleneck is in computing $\mathbf{X}\mathbf{X}_\mathbf{T}$ which demands $\Omega\left(d^2\right)$ operations per vector $\mathbf{x_t}$ (assuming they are dense), but in online PCA can strongly reduce time complexity.

According to equation 5, l is projection in l dimension donated by rule ii), OPT_k is optimal solution in k dimension in offline PCA, ε is a constant to compensate for the handicap of operating online and $\|\mathbf{X}\|_\mathrm{F}$ is the Frobenius norm of input data $\mathbf{X}$. In this paper, they built the relationship between l dimension and cost constant ε of online learning to find a sub-optimal solution to get close to optimal solution of l dimension.

$$\min_{\boldsymbol{\Phi}\in\mathcal{O}_{d,\ell}} \sum_{t=1}^{n} \|\mathbf{x}_t - \boldsymbol{\Phi}\mathbf{y}_t\|_2^2 \leq \mathrm{OPT}_k + \varepsilon\|\mathbf{X}\|_\mathrm{F}^2, \text{ where } \ell = \lceil 8k/\varepsilon^2\rceil \tag{5}$$