COMP6247(2020/21): Reinforcement and Online Learning Kalman and Particle Filters; Online PCA

Yan Zhang (yz10u20@soton.ac.uk)

1 SIS and Particle filter for the synthetic time varying AR problem

Kalman filter relies on the linearity and normality assumption. When the system is non-linear and the noise isn't Gaussian, the Kalman filter is not capable of solving the problem. Instead, when the model is non-linear and the noise isn't gaussian, Sequential Importance Sampling yields good results by approximating the true state using a weighted average of random draws from another proposal distribution.

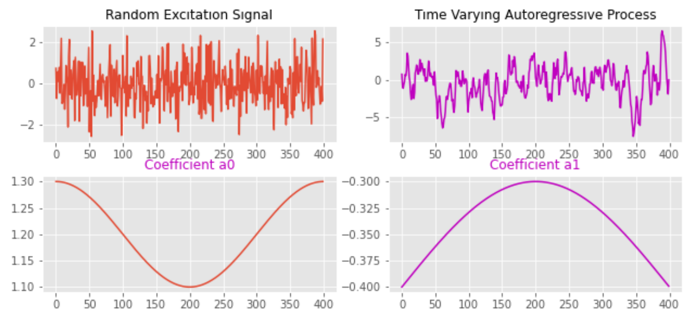

Figure 1: The synthetic time varying AR time series

In this section, SIS is applied for the synthetic time-varying AR problem, which is

$$s(n) = \sum_{k=1}^{2} a_k s(n-k) + v(n) \tag{1}$$

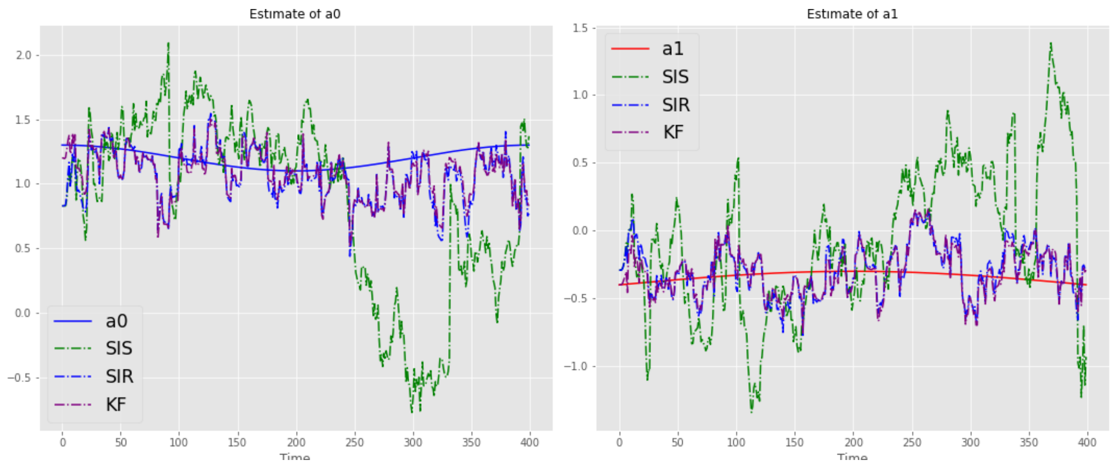

Figure 2: Estimates of parameters using Kalman Filter, SIS and SIR

The goal is estimating a_k, a 1×2 vector, sequentially. Figure 1 shows the AR time series problem. Unlike rejection sampling, SIS penalizes samples instead of rejecting samples by assigning a weight to each particle. The estimates of a_1 and a_2 using SIS are shown in Figure 2, denoted by the green dot line, and the error of the estimate is plotted in Figure 3. It is noted that the error increase with time, and the estimates are not accurate. As shown in the weight scatter plot in Figure 4, the weights of all particles are initially equal, but, as time goes, most of the particles' weights decrease to zero except for one. It is clear that SIS suffers from weight degeneracy, and only one particle could impact the estimates, leading to the inaccuracy in estimates.

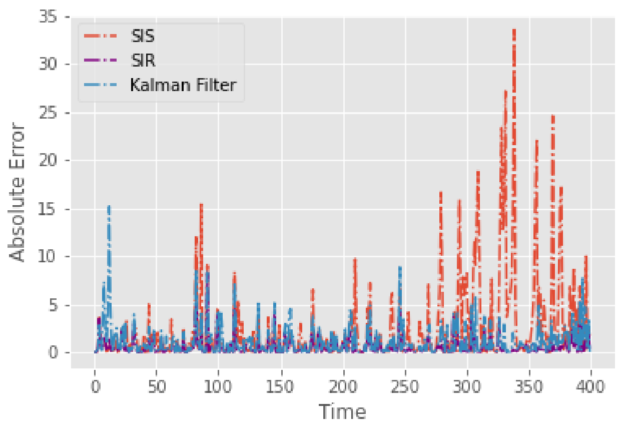

Figure 3: The error of estimates using Kalman Filter, SIS, and SIR

One approach to solving weight degeneracy in SIS is extending SIS to include a resampling step (SIR), which resizes each particle's weight to $1/N$, where N is the number of particles. The resampling step retains only stochastically relevant samples and helps reduce the variance. When SIR is used for filtering, it is called Particle Filter. The result of applying SIR to the synthetic time-varying AR time series problem is shown in Figure 2, denoted by a blue dot line. Figure 4 shows that the weights of all particles are equal along with the time index. Also, parameters estimated by the Kalman filter implemented in the second lab are denoted by a purple dot line. These two curves almost coincide, and the error plots of the SIR and Kalman filter are shown in Figure 3 are also similar. By comparison, the AR time series problem is linear and gaussian. Thus, SIR and Kalman Filter will produce the same likelihood. The critical difference between Particle Filter and Kalman Filter is that, instead of deriving analytic equations as the Kalman filter does, the Particle filter uses simulation methods to generate estimates of the state and the innovations [1]. In the case of linear and gaussian problems, Kalman Filter avoids simulations, which is more efficient, and we should use Kalman Filter rather than SIR. SIR is more suitable for the problem that the linearity and normality assumptions are not satisfied.

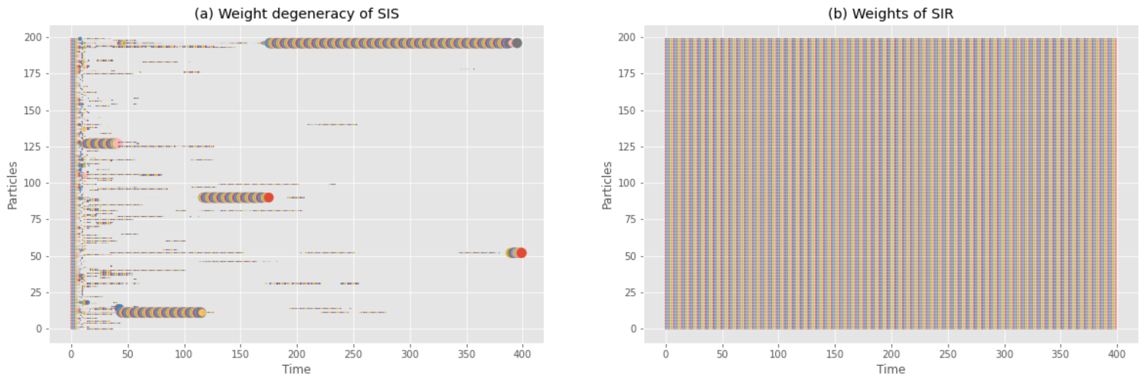

Figure 4: Weight degeneracy of SIS and weights of SIR

2 Extended Kalman Filter for logistic regression

Algorithm 1 Extended Kalman Filter for Logistic Regression [2]

```
1: function SIGMOID($x$)
2:     return $1/(1+e^{-x})$
3: end function
4: procedure EKF($X, y$)
5:     Initialization: $K, w, P, \theta$
6:     for $t \leftarrow 1$ to $N$ do
7:         $\hat{w}_{t|t-1} \leftarrow \hat{w}_{t-1|t-1}$
8:         $A_t \leftarrow SIGMOID(X\hat{w}_{t|t-1})(1 - SIGMOID(X\hat{w}_{t|t-1})X$
9:         $S_{t|t-1} \leftarrow SIGMOID(X\hat{w}_{t|t-1})$
10:        if $S_{t|t-1} > 0.5$ then
11:            $est_t \leftarrow 1$
12:        else
13:            $est_t \leftarrow 0$
14:        end if
15:        $u_{t|t-1} \leftarrow S_{t|t-1}(1 - S_{t|t-1})$
16:        $r_t \leftarrow S_{t|t-1}(1 - S_{t|t-1})$
17:        $K_t \leftarrow P_{t|t-1}A_t(A_t^T P_{t|t-1}A_t + r_t)^{-1}$
18:        $\hat{w}_{t|t} \leftarrow \hat{w}_{t|t-1} + K_t(y_t - S_{t|t-1})$
19:        $S_{t|t} \leftarrow SIGMOID(X\hat{w}_{t|t})$
20:        $u_{t|t} \leftarrow S_{t|t}(1 - S_{t|t})$
21:        $q_t \leftarrow \max(u_{t|t} - u_{t|t-1}, 0)$
22:        $P_{t|t-1} \leftarrow P_{t-1|t-1} + q_t I$
23:        $P_{t|t} \leftarrow P_{t|t-1} - K_t(A_t^T P_{t|t-1}A_t + r_t)K_t^T$
24:    end for
25:    return $w, K, est_t$
26: end procedure
```

In practice, the data are always noisy and non-linear. For a binary classification problem, the Logistic Regression classifier maps the data into two classes via a non-linear Sigmoid function. The model is not too non-linear and has Gaussian noise; thus, EKF yields good results. The Extended Kalman Filter is the most commonly used system for handling non-linear systems. By linearizing the non-linear function using the first two terms of its Taylor series expansion, we can approximate the non-linear function roughly. Algorithm 1 gives the pseudo-code of the EKF algorithm applied on Logistic Regression.

One approach to solving weight degeneracy in SIS is extending SIS to First, Generate synthetic two-class data with a distinct mean, common covariance, shown in Figure 5 (a), and apply EKF to the binary classification problem. Figure 5 (b) shows the convergence of the class boundary. We can see the speed of convergence is fast, as, at iteration 3, the decision boundary is close to the true solution. Figure 6(a) shows how the weights of the Logistic Regression classifier change over time. The two weights diverge into two distinct directions sharply in the beginning and afterward change symmetrically over time. These two weights will not converge to a stable value, but their ratio will remain unchanged. Figure 6 (b) noted that the local linearization made by EKF is quite efficient, and the accuracy of the linearization is high.

3 Particle Filter for Logistic Regression

The Particle Filter is also applicable for the sequential Logistic Regression problem. For the case of the binary classification problem, it is assumed that the data points generated in section 2 are shuffled and come one by

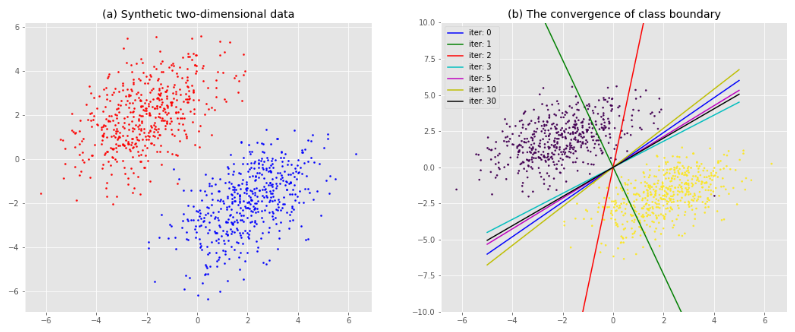

Figure 5: Graph (a) is the scatter plot of synthetic two-dimensional and two-class data generated from a distinct mean, common covariance; Graph (c) is how the decision boundary converges to the true solution, starting from a random guess.

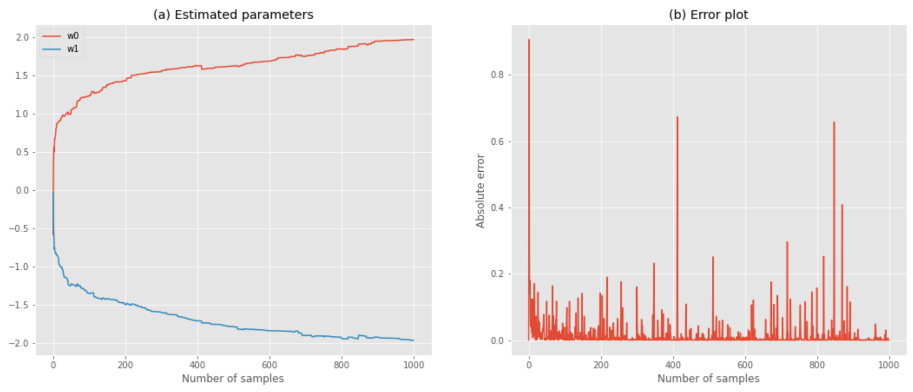

Figure 6: The plot of estimated parameters and error plot

one in time order. Algorithm 2 and 3 show the implementation of Particle Filter with a resampling step to avoid weight degeneracy. By comparing the convergence speed of the class boundary in Figure 5(b) and Figure 7(b), the Particle Filter converged much slowly than EKF, as it reaches near the true solution after iteration 500, while EKF reaches at iteration 3. Figure 7(a) shows the change of sequential logistic regression parameters, starting from an initial estimate. By comparing with Figure 6(a), I found Particle Filter updates parameters much slower than EKF; thus, Particle Filter takes more time before convergence.

Table 1: Measurements for the efficiency of EKF and Particle Filter

Methods	Run time	Accuracy score	Averaged estimate error
EKF	0.066s	0.997	0.016
Particle Filter	155.5s	0.998	0.011

Table 1 shows the efficiency of EKF and Particle Filter. The most conspicuous thing is that Particle Filter takes 2000 times longer than EKF, which is relatively inefficient. While Particle Filter takes a long time to run, the accuracy score only increases EKF by 0.001, which is 98.8%. Both of the accuracy scores are high, and the average estimated error is low, 0.016 for EKF and 0.011 for Particle Filter.

Algorithm 2 Resampling[3]

```
1: procedure Resampling($\{(w_0^{(1)}, \theta_0^{(1)}), ..., (w_0^{(L)}, \theta_0^{(L)})\}$)
2:     Initialize the CDF: $c_1 = 0$, and $\hat{\theta}_t = \emptyset$
3:     for $i \leftarrow 1$ to $L$ do
4:         Sample $c_l = c_{l-1} + w_t^{(l)}$
5:     end for
6:     Start at the bottom of the CDF: $i = 1$
7:     Draw a starting point: $u_1 \sim \mathcal{U}(0, \frac{1}{L})$
8:     for $i \leftarrow 1$ to $L$ do
9:         Evaluate thresholds: $u_{l+1} = u_l + \frac{1}{L}$
10:        while $u_l > c_i$ do
11:            $i = i + 1$
12:        end while
13:        Append Samples: $\hat{\theta}_t = (\hat{\theta}_t, \theta_t)$
14:    end for
15:    return $\{(\frac{1}{L}, \hat{\theta}_t^{(1)}), ..., (\frac{1}{L}, \hat{\theta}_t^{(L)})\}$
16: end procedure
```

Algorithm 3 Particle Filter for Logistic Regression[3]

```
1: procedure PF($X, y, \sigma_y^2, \sigma_\theta^2$)
2:     Initialization: $\{(w_t^{(1)}, \theta_t^{(1)}), ..., (w_t^{(L)}, \theta_t^{(L)})\}_{t=1}^2$
3:     for $t \leftarrow 3$ to $T$ do
4:         $X_t \leftarrow [X[t-1], X[t-2]]$
5:         for $l \leftarrow 1$ to $L$ do
6:             Sample $\theta_t^{(l)} \sim \mathcal{N}(\theta_t | \theta_{t-1}^{(l)}, \sigma_\theta^2)$
7:             Append $\theta_{1:t}^{(l)} = (\theta_{1:t-1}^{(l)}, \theta_t^{(l)})$
8:             Calculate probability $p_t^l = \mathcal{N}(y_t | SIGMOID(X_t \theta_{t-1}^{(l)}), \sigma_y^2)$
9:             Assign the particle a weight $w_t^{(l)} = w_{t-1}^{(l)} p_t^l$
10:        end for
11:        for $l \leftarrow 1$ to $L$ do
12:            Normalise $w_t^{(l)} = w_t^{(l)} / \sum_{l'=1}^{L} w_t^{(l')}$
13:        end for'
14:        Calculate $\hat{Neff} = \frac{1}{\sum_{i=1}^{L}} (w_t^{(l)})^2$
15:        if $\hat{Neff} < L$ then
16:            Resampling using Algorithm 2
17:        end if
18:    end for
19:    return $\{(\frac{1}{L}, \theta_t^{(1)}), ..., (\frac{1}{L}, \theta_t^{(L)})\}_{t=1}^T$
20: end procedure
```

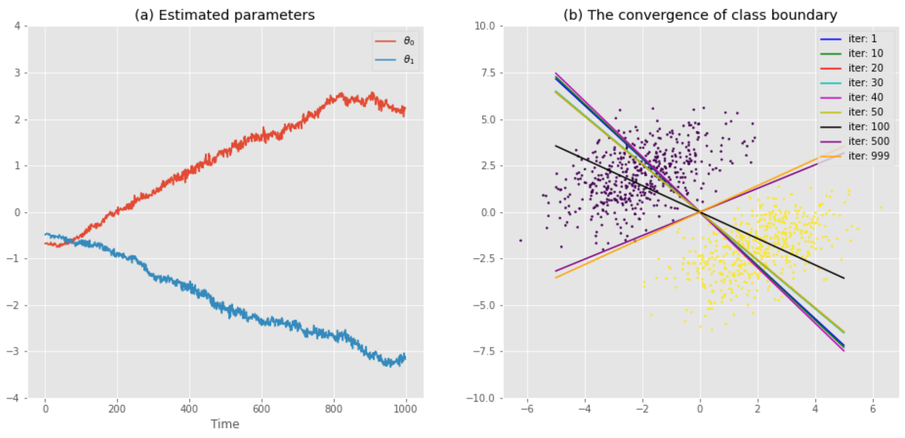

Figure 7: (a)The plot of parameters of the sequential Logistic Regression problem using Particle Filter and (b)the convergence of the class boundary

4 Summary

Particle Filter follows the basic idea of Markov assumption (assuming that the current state is only base on the previous one) and works for any arbitrary distribution and not just Gaussian. Like Kalman Filter, Particle Filter employs the previous state to estimate the current state, and it is therefore called a filter. When the model is linear, and the noise is Gaussian, Kalman Filter is the best choice. EKF is the best model when the problem is not too non-linear and has Gaussian noise. The Particle Filter is suitable for any arbitrary distribution and not just Gaussian but at the expense of high computational power.

References

[1] J. Fernández-Villaverde, *Kalman and particle filtering*. London: Palgrave Macmillan UK, 2017.

[2] S. M. Lee and S. J. Roberts, "Sequential dynamic classification using latent variable models," *The Computer Journal*, vol. 53, no. 9, pp. 1415–1429, 2010.

[3] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking," *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp. 174–188, 2002.