COMP6247 Lab 3: Kalman and Particle Filters; Online PCA

Wei Chien Teoh (Eugene)
wct1c16@soton.ac.uk

16 April 2021

1 Introduction

This report presents the findings and results for Lab 3 of COMP6247 [1]. The code implementation is stored in a Github repository [2].

2 Task 1

Task 1 presents my implementation of the Sequential Importance Sampling (SIS) and Sequential Importance Resampling (SIR). The results of SIS and SIR will be compared to the Kalman Filter (KF) implementation [3] for the state estimation of the time-varying autoregressive time series problem stated in Lab 2 [4].

A second-order time-varying autoregressive time series identical to [4] is generated with a time index, T of 200.

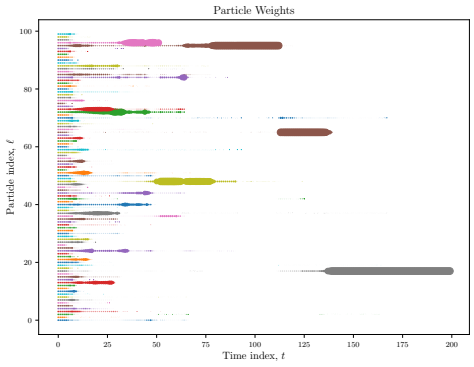
Initially, the Sequential Importance Sampling (SIS) algorithm described in [5] was implemented. However, due to the recursive multiplication of weights, almost all particle weights becomes infinitesimally small except for one. This behaviour is shown in Fig. 1a, where initially the size of the weights are equal. After several iterations, the weights are mostly focused on only one particle. This phenomenon is known as weight degeneracy. This signifies that the posterior distribution is majorly dependent on only one particle.

To solve weight degeneracy, an extra resampling step [5] is added to the original SIS algorithm, which is then known as Sequential Importance Resampling (SIR). For every iteration, the weights are resampled to have equal sizes, shown in Fig. 1b. Fig. 2 shows the Effective Sample Size (ESS) before and after introducing the resampling step. With resampling introduced, larger number of particles are effectively accounted to the estimation of the posterior PDF.

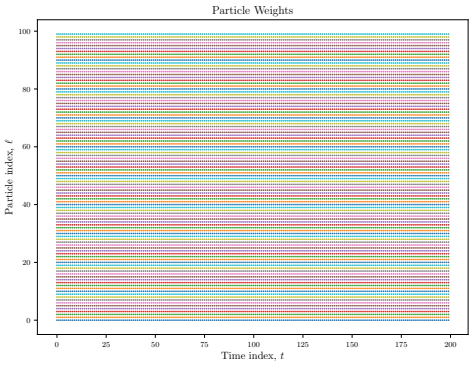
However, resampling at every time step may cause the particles to come from the same ancestor particle, thus causing path degeneracy [6]. Rather than resampling at every time step, one could resample when the ESS reduces to a certain threshold. In this experiment, a threshold of 60% was set. This signifies that the resample will occur when the ESS reduces below 60% of the total number of particles N_s.

Fig. 3 shows the estimation of parameters a_0 and a_1 of the second-order time-varying autoregressive time series using a Kalman Filter implemented in [3], SIS, SIR and SIR with threshold. The number of particles, N_s is set to 100. It is shown that all four algorithms did not provide "smooth" estimations. This is potentially due to the original process model described in [4] being a random walk model instead of being non-linear. The Mean Squared Error (MSE) calculated for the four algorithms implies that the performance is in the order: SIR with threshold, SIR, KF, SIS (best to worse). SIR is known to be able to solve non-linear state estimation. The results have proven the hypothesis.

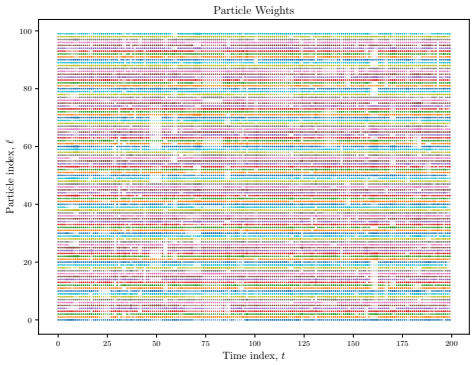
Sequential Monte Carlo methods (SIS and SIR) comes with an expense of computation time due to the repeated sampling. The average runtime for the three algorithm was calculated using %timeit in Python and stated in the caption of Fig. 3. SIS and SIR used almost 9 times as much time for computation. This is potentially an issue to be considered in high dimensional state space models.



(a) SIS



(b) SIR



(c) SIR with Threshold

Figure 1: Plot of particle weight size at each time step. The diameter of markers scales with the size of particle weights.

3 Task 2

Task 2 presents derivations and results of the Extended Kalman Filter (EKF) algorithm for the logistic regression problem described in the Lab 3 exercise sheet. Extended Kalman Filter allows state estimation of non-linear Gaussian state-space models by approximating the non-linear function as its Taylor approximation to the first order. Non-linearity can be present in either or both state and observation models.

Following the exercise sheet, consider we have a simple binary classification problem to be solved

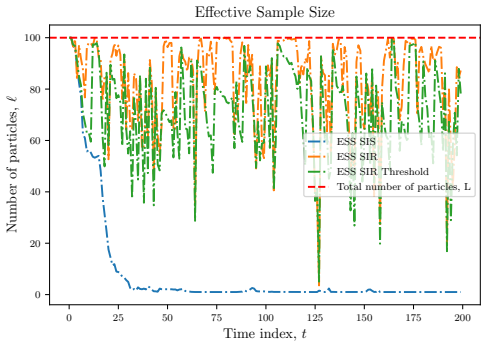


Figure 2: Effective sample size of SIS and SIR algorithm.

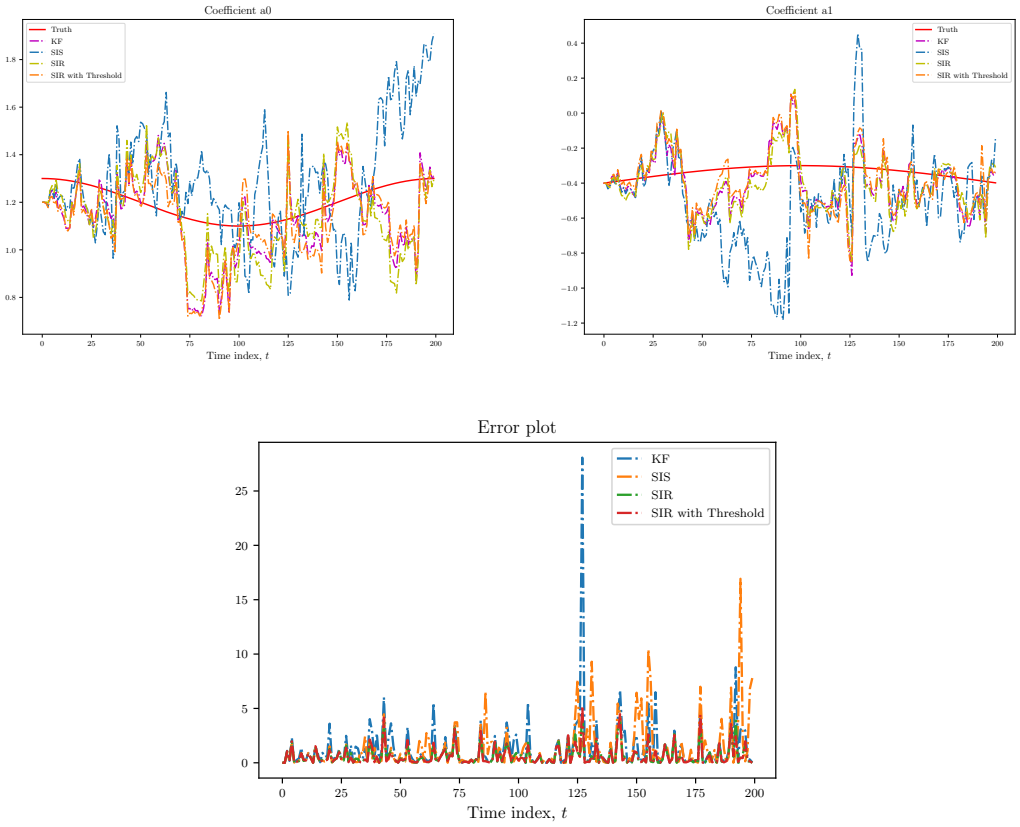


Figure 3: Plots of unknown parameters a_0 and a_1 estimated using KF, SIS and SIR, along with its squared errors. Mean Squared Error: (KF) 1.2610 (SIS) 1.3363 (SIR) 0.6150 (SIR with Threshold) 0.6071; Average Runtime: (KF, 100 loop average) 0.534s (SIS, 5 loop average) 4.65s (SIR, 5 loop average) 4.68s

sequentially using a logistic regression. We have a state space model:

$$\begin{aligned} \boldsymbol{\theta}(n) &= \boldsymbol{\theta}(n-1) + \boldsymbol{w}(n) \\ y(n) &= f(\boldsymbol{\theta}(n), \boldsymbol{x}_n) + v(n), \end{aligned} \tag{1}$$

with the non-linear function as the logistic regression:

$$\begin{aligned} f(\boldsymbol{\theta}(n), \boldsymbol{x}_n) &= g(\boldsymbol{\theta}(n)^T \boldsymbol{x}_n) \\ g(z) &= \frac{1}{1+e^{-z}}, \end{aligned} \tag{2}$$

As mentioned above, Eq. (2) can be approximated by it's first-order Taylor approximation:

$$f(\boldsymbol{\theta}(n), \boldsymbol{x}_n) \approx f(\boldsymbol{\theta}(n|n-1), \boldsymbol{x}_n) + \hat{\boldsymbol{F}}_n^T (\boldsymbol{\theta}(n) - \boldsymbol{\theta}(n|n-1)) \tag{3}$$

where the Jacobian is described as:

$$\hat{\boldsymbol{F}}_n = \hat{\boldsymbol{F}}(\boldsymbol{\theta}(n|n-1), \boldsymbol{x}_n) = g(\boldsymbol{\theta}(n|n-1)^T \boldsymbol{x}_n)(1 - g(\boldsymbol{\theta}(n|n-1)^T \boldsymbol{x}_n))\boldsymbol{x}_n \tag{4}$$

With the approximation above, the Extended Kalman Filter equations for the logistic regression problem are:

$$\begin{aligned} \boldsymbol{\theta}(n|n-1) &= \boldsymbol{\theta}(n-1|n-1) \\ P(n|n-1) &= P(n-1|n-1) + Q \\ e(n) &= y(n) - f(\boldsymbol{\theta}(n|n-1), \boldsymbol{x}_n) \\ \boldsymbol{\theta}(n|n) &= \boldsymbol{\theta}(n|n-1) + \boldsymbol{k}(n) e(n) \\ P(n|n) &= (I - \boldsymbol{k}(n)\hat{\boldsymbol{F}}_n) P(n|n-1) \\ \boldsymbol{k}(n) &= \frac{P(n|n-1)\hat{\boldsymbol{F}}_n^T}{R + \hat{\boldsymbol{F}}_n P(n|n-1)\hat{\boldsymbol{F}}_n^T} \end{aligned} \tag{5}$$

Eq. (5) follows the same notation as in [1].

The EKF algorithm described in Eq. (5) will be utilised to attempt to solve two binary classification problems with different α, illustrated in Fig. 4. The dataset is generated using Numpy following the instructions in [1].

Fig. 5a shows the error plot of both problems. It is shown that an increase in α allows for a better convergence of error. When α decreases, the problem becomes non-linearly separable. Hence, with $\alpha = 0.5$, the convergence of error is not possible with a linear classifier such as logistic regression. The best accuracy is calculated by taking the $\boldsymbol{\theta}$ of the time step of the lowest error.

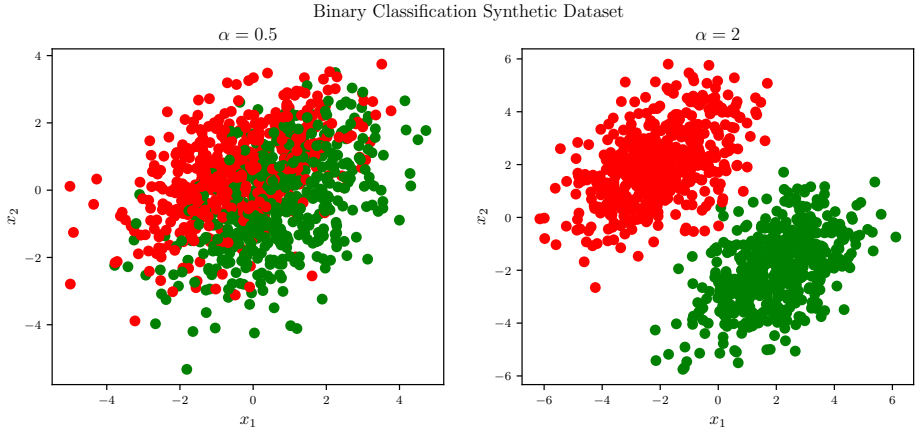
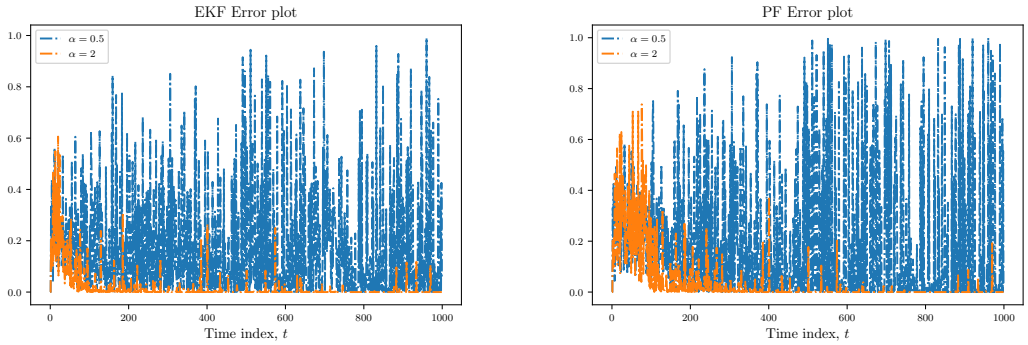


Figure 4: Binary classification synthetic dataset with different mean α.



(a) EKF
Best accuracy: ($\alpha = 0.5$) 76.2% ($\alpha = 2$) 99.8%

(b) PF; $N_s = 100$
Best accuracy: ($\alpha = 0.5$) 75.6% ($\alpha = 2$) 99.6%

Figure 5: Logistic regression squared error with different α.

4 Task 3

Task 3 presents the pseudocode and results of the Particle Filter (PF) algorithm for the logistic regression problem described in Section 3. In Section 2, we discussed the superiority of the resampling step when added to the original SIS algorithm. Hence, in this section, the PF algorithm will include the resampling step.

Eq. (6) shows the probabilistic representation of the state space model described in Eq. (1). The assumption is made that both state and observation models are Gaussian. Algorithm 1 describes the particle filtering algorithm for the logistic regression problem. Algorithm 1 follows the same notation as in [6].

$$\begin{aligned} p(\boldsymbol{\theta}(n)|\boldsymbol{\theta}^i(n-1))) &= \mathcal{N}(\boldsymbol{\theta}(n)|\boldsymbol{\theta}(n-1), Q) \\ p(y(n)|\boldsymbol{\theta}^i(1:n)) &= \mathcal{N}(y(n)|f(\boldsymbol{\theta}(n), \boldsymbol{x}_n), R) \end{aligned} \tag{6}$$

Algorithm 1: Particle Filter for Logistic Regression

Initialise: $\{(w^1(0), \boldsymbol{\theta}^1(0)), \cdots, (w^{N_s}(0), \boldsymbol{\theta}^{N_s}(0))\}$
Output: $\{(w^1(n), \boldsymbol{\theta}^1(n)), \cdots, (w^{N_s}(n), \boldsymbol{\theta}^{N_s}(n))\}_{n=1}^{N}$
for $n = 1, \cdots, N$ **do**
  for $i = 1, \cdots, N_s$ **do**
    Sample $\boldsymbol{\theta}^i(n) \sim p(\boldsymbol{\theta}(n)|\boldsymbol{\theta}^i(n-1)))$;
    Append $\boldsymbol{\theta}^i(1:n) = (\boldsymbol{\theta}^i(1:n-1), \boldsymbol{\theta}^i(n))$;
    Evaluate $w^i(n) = w^i(n-1)p(y(n)|\boldsymbol{\theta}^i(1:n))$;
  end
  for $i = 1, \cdots, N_s$ **do**
    Normalise $w^i(n) = w^i(n)/\sum_{j=1}^{N_s} w^j(n)$;
  end
  Resampling;
end

Fig. 5b shows the error plot of the particle filter. Both algorithms show similar results. However, the particle filter has more fluctuations of error. This is due to the small sample size N_s that was used for the algorithm. Increasing N_s will allow less fluctuations at the expense of computation time. The accuracy of classification for both algorithms has proved to be nearly identical.

Note that both EKF and PF are non-optimal filters due to the approximations. However, the advantages of PF is that it is able to work with non-linear non-Gaussian state space models. PF converges to the optimal solution when number of particles N_s approaches infinity. The EKF derivations assumes Gaussianity in the state space model, but is able to solve non-linear problems.

References

[1] Mahesan Niranjan and Christine Evers. *COMP6247(2020/21): Reinforcement and Online Learning – Kalman and Particle Filters; Online PCA*. School of Electronics and Computer Science, University of Southampton, 19 April 2021.

[2] Eugene Teoh. *COMP6247 Labs*. 19 April 2021. URL: `https://github.com/eugeneteoh/COMP6247-Labs`.

[3] Eugene Teoh. *COMP6247 Lab 2: Kalman Filter*. School of Electronics and Computer Science, University of Southampton, 4 March 2021.

[4] Mahesan Niranjan. *COMP6247(2020/21): Reinforcement and Online Learning – Kalman Filter*. School of Electronics and Computer Science, University of Southampton, 16 February 2021.

[5] M. S. Arulampalam et al. "A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking". In: *IEEE Transactions on Signal Processing* 50.2 (2002), pp. 174–188. DOI: `10.1109/78.978374`.

[6] Christine Evers. *COMP6247 Week 6: Sequential Monte Carlo Lecture Slides*. Mar. 2021.