Bohao Yang
31799256
by2u20@soton.ac.uk

Reinforcement and Online Learning Lab 3 Extended Kalman Filter and Particle Filter

University of Southampton

2021-04-17

1 Implement SIS, SIR and Kalman Filter for Synthetic time varying AR process

1.1 Synthetic time varying AR process

An autoregressive time series(AR Process) of order p is given by the generating model as below:

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

where n is index over time, $a_k$are the parameters of process and $v(n)$is Gaussian random noise. Figure 1 shows the a random excitation signal and a time varying AR process and its coefficients.

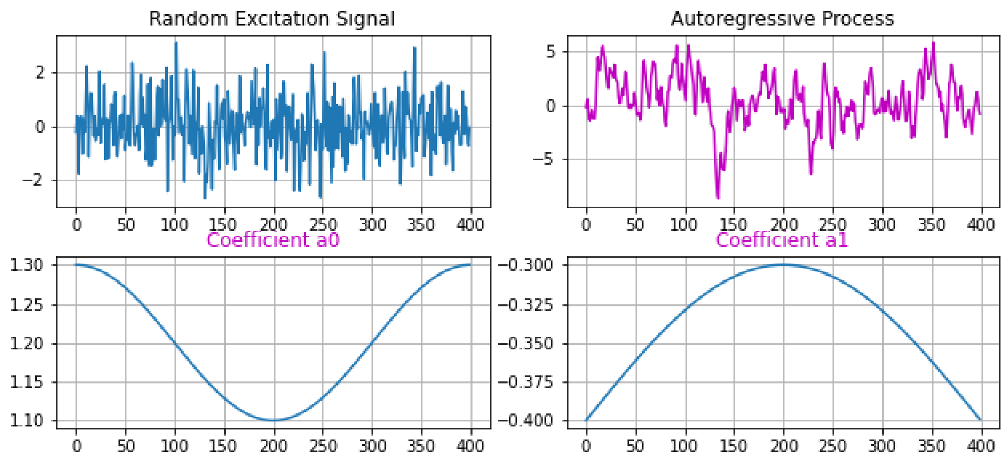

Figure 1: random excitation signal and synthetic time varying AR process

The state space model for sequential estimation of parameters of a time series model is as follows:

$$\theta(n) = f(\theta, n-1) + w(n)$$

$$y(n) = \theta^T x_n + v(n)$$

where we have assumed a random walk model on the unknown parameters θ, a univariate observation and 2 past samples are held in the vector $x(n) = [x(n-1), x(n-2)]$. At every time step, our task is to make the best estimate θ from observation $y(n)$ and input $x(n)$.

1.2 SIS for AR process

First initialising 30 samples from Gaussian distribution as the estimated parameters and their corresponding weights, their distributions are shown in Fig 2.
At each time step, the samples are updated according to state space model shown above. By simply adding an additional beta times Gaussian noise to the samples calculated from the previous time step, the beta was set to 0.01. Weights are updated according to the forums in Algorithm 1 SIS pseudocode below.

As can be seen from the changes of SIS weights in Fig 2, most weights will be equals zero at the end from nearly equally distributed at the beginning. The performance of SIS for estimating parameters for the AR process is terrible according to the last two plots in Fig 2, the orange curve represents the actual parameters, and blue ones are estimated ones.

Algorithm 1 SIS for AR process

```
Input: beta, $\sigma_\theta^2$, $\sigma_y^2$, $y_{1:t}$, t = 1,..., T
Initialization: $\theta_0^\ell$, $w_0^\ell$, $\ell$ =1, ..., Ns
for t=2,...,T do
    for $\ell$=1,...,Ns do
        $\theta_t^{(\ell)} \sim N\left(\theta_t \mid \theta_{t-1}^{(\ell)}, beta * \sigma_\theta^2\right)$
        $x = [y_{t-1}, y_{t-2}]$
        $p(y_t \mid \theta_t^{(\ell)}) \sim N(_t| \theta_t^{(\ell)T} x, \sigma_y^2)$
        $w_t^{(\ell)} = w_{t-1}^{(\ell)} p(y_t \mid \theta_t^{(\ell)})$
    end
    for $\ell$=1,...,Ns do
        $\tilde{w}_t^{(\ell)} = w_t^{(\ell)} / \sum_{\ell'=1}^{L} w_t^{(\ell')}$
    end
end
```

1.3 SIR for AR process

In order to tackle with weight degeneracy problem in SIS method, SIR add a resampling step to the end of SIS method so that the weights of each sample will be equal.
The process of resampling has been shown in Algorithm 2 pseudocode below.
Fig 3 shows that weights of samples are equally distributed from the beginning to the end of the whole process. Besides, the last two plots in Fig 3 indicate that the performance of estimating parameter using SIR is better than that of SIS.

Algorithm 2 Resampling Algorithm

```
Input: samples $\theta_0^\ell$, weights $w_0^\ell$, $\ell$ from 1 to Ns
Initialization: cumulative density function (CDF) of weights: $c_1$ = 0, i=1
for $\ell$=1,...,Ns do
    Sample $c_\ell = c_{\ell-1} + w^{(\ell)}$ (CDF)
end
Sample $u_1 \sim U]0, \frac{1}{Ns}]$
for $\ell = 1, \ldots, Ns$ : do
    Evaluate thresholds: $u_{\ell+1} = u_\ell + \frac{1}{Ns}$
end
for $\ell = 1, \ldots, Ns$ : do
    while $u_\ell > c_i$ do
        Set $i = i + 1$
    end
    Out: $\left\{\left(\frac{1}{Ns}, \theta_t^{(1)}\right) \ldots, \left(\frac{1}{Ns}, \theta_t^{(Ns)}\right)\right\}$
end
```

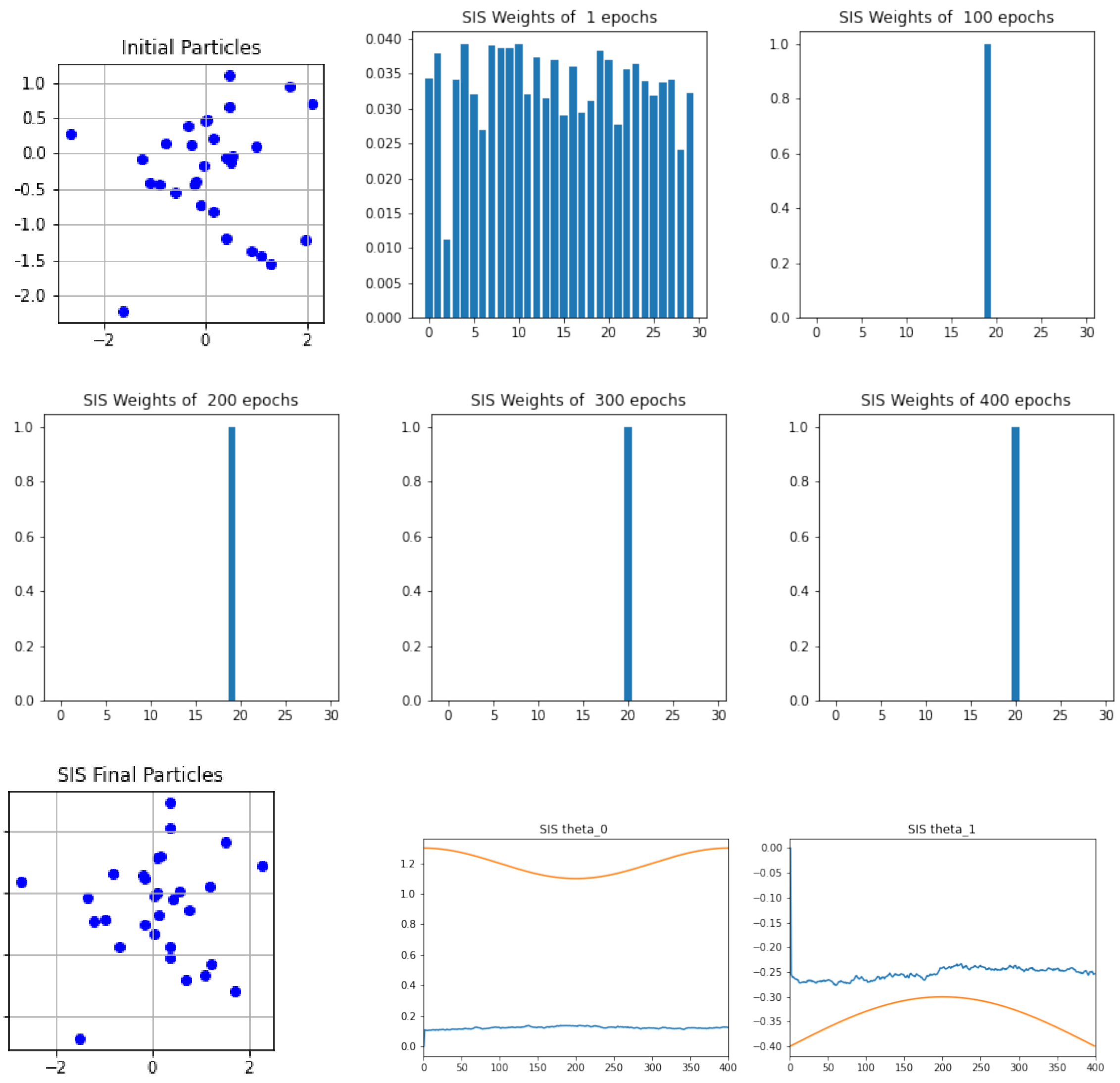

Figure 2: Weights and parameters of AR process estimated by SIS

1.4 Kalman Filter for AR process

Fig 4 shows the result of estimated parameters using Kalman Filter setting R equals 100.838 and Q equals the identity matrix (I) and Fig 5 shows the comparision between true and predicted value of AR process in each time step by SIR and Kalman Filter and MSE error, respectively.
It is evident that the MSE of Kalman Filter is smaller than that of SIR, and by comparing the predicted and true values of parameters and observation, Kalman Filter performed better than SIR in estimating parameters of time varying AR process. The reason is that time varying AR process is a linear system and parameters that can be estimated well by Kalman Filter. As for SIR, its performance partly depends on the number of samples and initial value of samples. Samples here means the parameters of AR process.
In conclusion, Kalman Filter is suitable for linear system parameter estimation but SIR is a method which provide approximation solution which is not stable each time (influenced by number and inital value of sample).

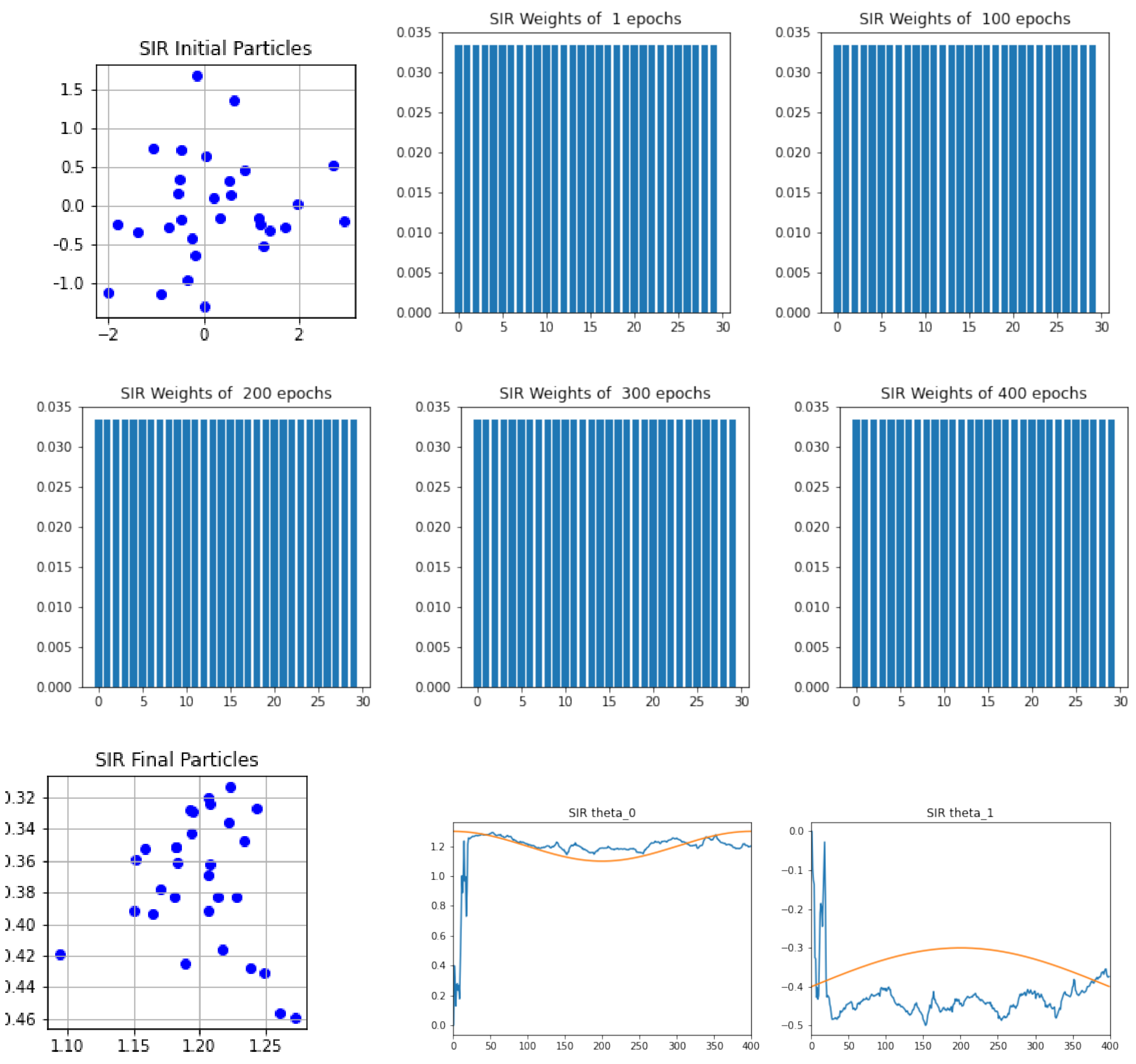

Figure 3: Weights and parameters of AR process estimated by SIR

2 Implement Extended Kalman Filter (EKF) algorithm for the logistic regression problem

First generating two dimensions data in which features are Gaussian distributed:
$m_1 = \begin{bmatrix} -1 & 1 \end{bmatrix}$ and $m_2 = \begin{bmatrix} -1 & 1 \end{bmatrix}$ $C_1 = C_2 = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$
m_1 and m_2 represent the mean and C_1 and C_2 represent the covariance matrices of two classes data.
Now assuming a state space model is as follows:

$$\theta(n) = \theta(n-1) + w(n)$$

$$y(n) = f(\theta, x_n) + v(n)$$

where$f(.,.)$is a nonlinear function and in logistic regression problem can be represented as:

$$f(.,.) = \frac{1}{1 + exp(-\theta^T x)}$$

The initial parameters of logistic regression θ are set randomly, and the decision boundary formed by it is shown in the second plot in Fig 6. When implementing Extended Kalman Filter described as the pseudocode in Algorithm 3, hyperparameters are setting according to Table 1.

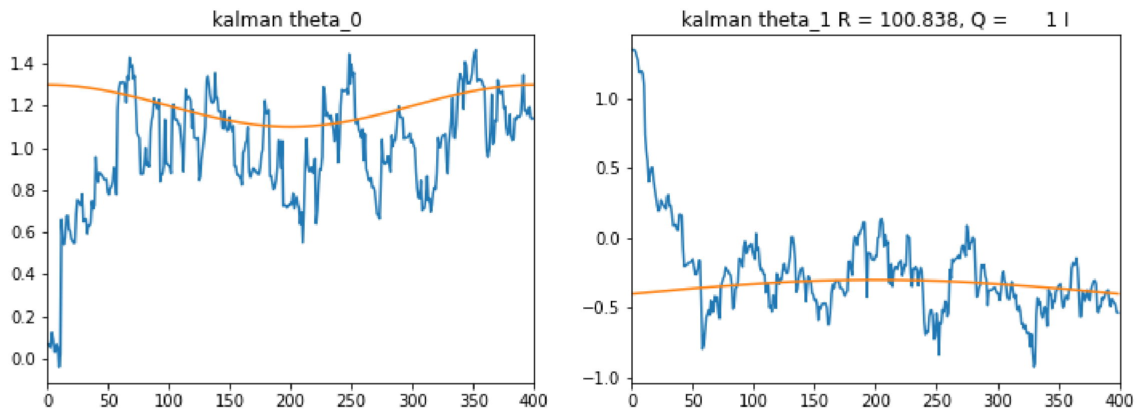

Figure 4: Estimated and true parameters of AR process by Kalman Filter

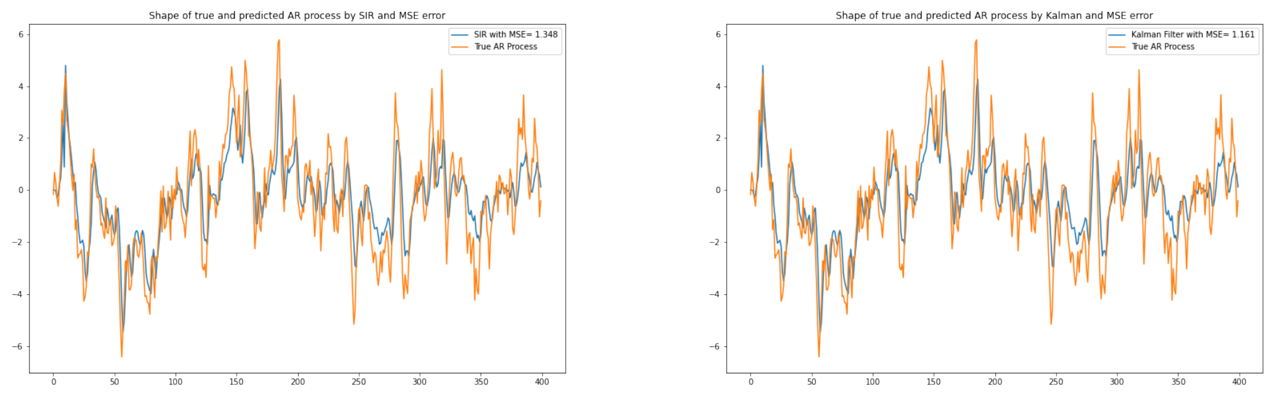

Figure 5: Shape and MSE error of true and predicted AR process by SIR and Kalman

The change of decision boundary after each 100 time steps can be seen in Fig 6 and till the end of the EKF process, the decision boundary can converge to an ideal result; only a few data points are misclassified.

Table 1: Hyperparameters and Value for EKF

Hyperparameters	value
length of data T	400
number of samples Ns	30
beta	0.01
σ_θ^2	beta * 2 dimension vector from a uniform distribution
σ_y^2	standard deviation of a T length random Gaussian noise from a uniform distribution

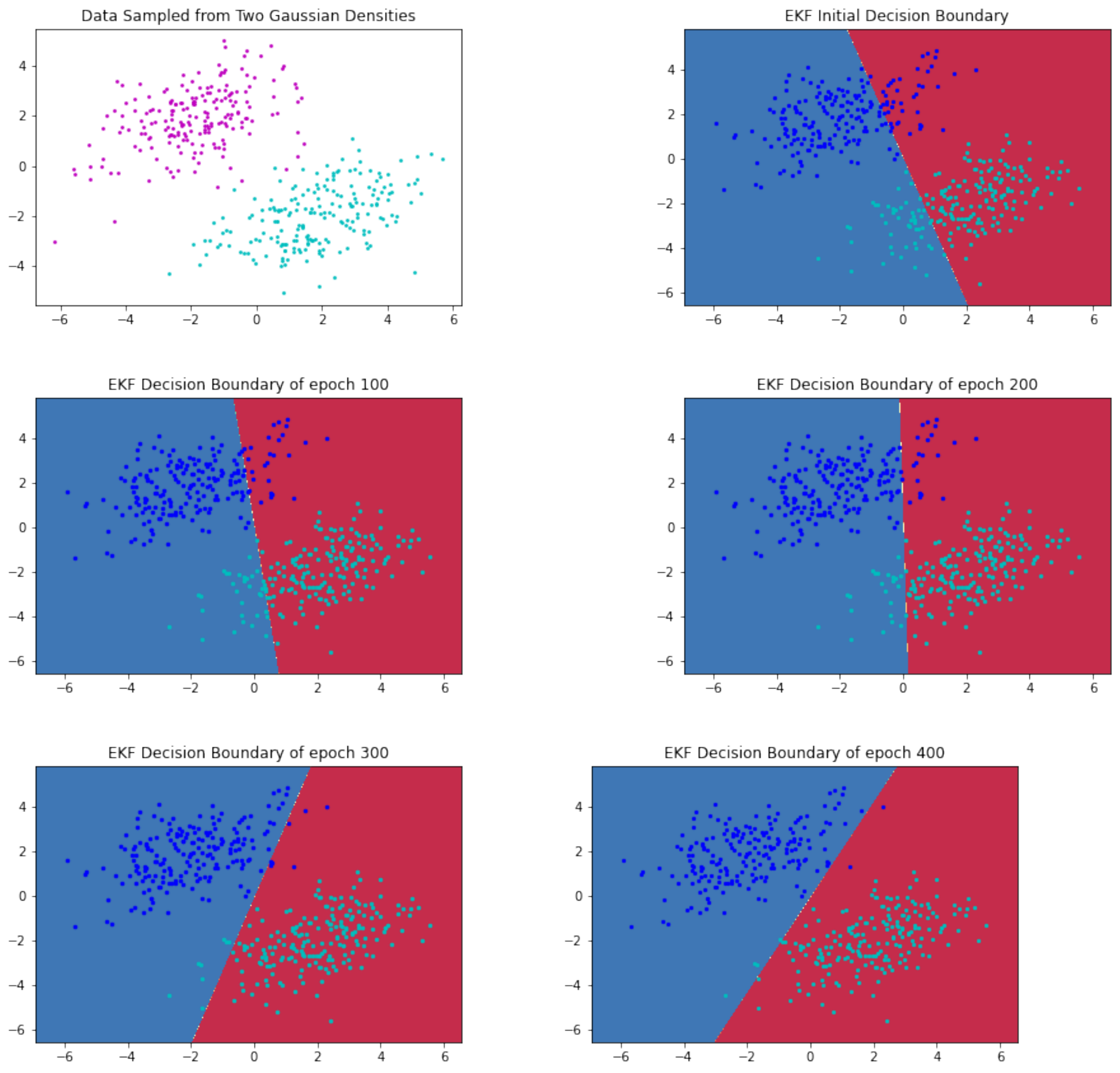

Figure 6: Data distribution and decision boundary estimated by EKF

Algorithm 3 EKF for logistic regression

```
Input: beta, $\sigma_\theta^2$, $\sigma_y^2$, $x$, $y_{1:t}, t = 1, \dots, T$
Initialization: $\theta_0$
for t=1,...,T do
    $x = x_{1:t}$
    Prediction:
    $\theta_{t|t-1} = \theta_{t-1}$
    $\Sigma_{t|t-1}$ = beta*$\sigma_\theta^2$ + $\Sigma_{t-1}$
    $H = \frac{1}{1+\exp\left(-\boldsymbol{\theta}^T\boldsymbol{x}\right)} * (1 - \frac{1}{1+\exp\left(-\boldsymbol{\theta}^T\boldsymbol{x}\right)}) * (-x)$
    $e = y_n - \frac{1}{1+\exp\left(-\boldsymbol{\theta}^T\boldsymbol{x}\right)}$
    $K = \frac{\Sigma_{t|t-1}H}{H\Sigma_{t|t-1}H^T+\sigma_y^2}$
    Update:
    $\theta_t = \theta_{t|t-1} + Ke$
    $\Sigma_t = \Sigma_{t|t-1} - KH\Sigma_{t|t-1}$
end
```

3 Particle Filter for sequential logistic regression problem

Now considering solving logistic regression problem with Particle Filter (pseudocode of Algorithm 4). Setting the length of data T, beta, Ns, σ_θ^2 and σ_y^2 equal to that in EKF algorithm and implementing Particle Filter, Fig 7 shows the distributions of data, particles and weights, initial decision boundary and the change of decision boundary formed by estimated parameters every 100 time steps. The initial decision boundary is formed by randomly selected parameters.

Till the end of Particle Filter process, as shown in Fig 7, there are no misclassification data points. Therefore, the performance of Particle Filter is better than that of EKF for logistic regression problem in the situation of hyperparameters set according to Table 1.

Extended Kalman Filter bases on a local linearisation of the equations is a sufficient description of nonlinearity. Specifically, it utilises the first term of Taylor expansion of the nonlinear function and approximates the likelihood equation $p(y_t \mid \theta_t)$ to be Gaussian.

Sequential Importance and Resampling (Particle Filter) draw a group of samples from proposal distribution (prior distribution of θ) each time step and assign weights to them. The performance of estimated parameters will be influenced by the numbers of samples, Ns and initial value of samples, to some extent. After several experiments, it is found that 30 is the proper number for Ns, neither too few to describe the parameters nor too many to increase the computational consumption.

Another thing about Paticle Filter is that if the initial decision boundary can classify much data accurately, as shown in Fig 7, it will not take too many iterations to converge an optimal solution. On the other hand, if the initial values that formed decision boundary are not ideal, Particle Filter might not converge to the parameter values that form the true decision boundary.

Algorithm 4 SIR for logistic regression

```
Input: beta, $\sigma_\theta^2$, $\sigma_y^2$, $x$, $y_{1:t}$, t = 1,. . . , T
Initialization: $\theta_0^\ell$, $w_0^\ell$, $\ell$ = 1 ,. . . , Ns
for $t$=1,...,T do
    for $\ell$=1,...,Ns do
        $x = x_{1:t}$
        $\theta_t^{(\ell)} \sim N\left(\theta_t \mid \theta_{t-1}^{(\ell)}, beta * \sigma_\theta^2\right)$
        $p(y_t \mid \theta_t^{(\ell)}) \sim N(y_t \mid \frac{1}{1+\exp\left(-\boldsymbol{\theta}^T \boldsymbol{x}\right)}, \sigma_y^2)$
        $w_t^{(\ell)} = w_{t-1}^{(\ell)} p(y_t \mid \theta_t^{(\ell)})$
    end
    for $\ell$=1,...,Ns do
        $\tilde{w}_t^{(\ell)} = w_t^{(\ell)} / \sum_{\ell'=1}^{L} w_t^{(\ell')}$
    end
    Resampling(Algorithm 2)
end
```

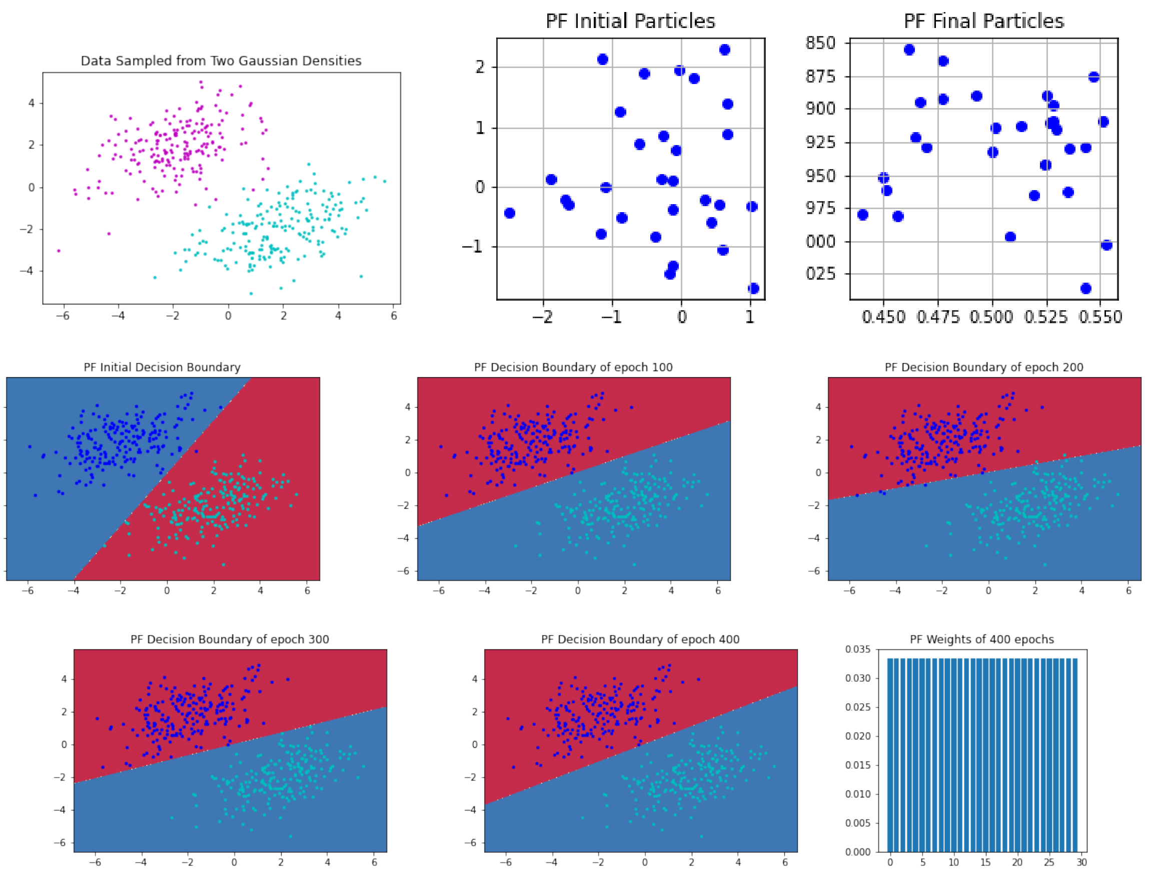

Figure 7: Partical filter for sequential logistic regression