STAT 576 Bayesian Analysis

Lecture 11: State-space Models and Sequential Monte Carlo II

Chenceng Cai

Washington State University

Sequential Monte Carlo

- Last time, we introduced the state-space models.
- For linear Gaussian state-space models, we can use Kalman filter and smoother to estimate the latent states and parameters.
- The key idea behind the Kalman filter and smoother is to recursively update the filtering and smoothing distributions.
- For general state-space models, we usualy do not have closed-form solutions as in the linear Gaussian case.
- Sequential Monte Carlo (SMC) methods provide a general framework for estimating the filtering and smoothing distributions in general state-space models through Monte Carlo sampling.

The Sequential Structure (MC version)

- In our previous discussion for the Kalman filter and smoother, we have the following recursive structure:

$$X_t \mid \boldsymbol{Y}_t \sim \mathcal{N}(\boldsymbol{\mu}_t, \boldsymbol{V}_t) \implies X_{t+1} \mid \boldsymbol{Y}_t \sim \mathcal{N}(\boldsymbol{\mu}_{t+1}, \boldsymbol{V}_{t+1}).$$

  It is a consequence of the fact that $(X_{t+1}, Y_{t+1}) \mid X_t$ is multivariate normal.

- (**MC version**) Similarly, if we have samples $(\boldsymbol{X}_t^{(i)}, w_t^{(i)})_{i=1}^N$ from the filtering distribution $p(\boldsymbol{X}_t \mid \boldsymbol{Y}_t)$, we can generate samples from the filtering distribution $p(\boldsymbol{X}_{t+1} \mid \boldsymbol{Y}_{t+1})$ by the following steps:
  1. Sample $X_{t+1}^{(i)} \sim q_{t+1}(X_{t+1})$ for some proposal distribution q_{t+1}
  2. Let $\boldsymbol{X}_{t+1}^{(i)} = (\boldsymbol{X}_t^{(i)}, X_{t+1}^{(i)})$ and assign weights

$$w_{t+1}^{(i)} = w_t^{(i)} \frac{f_{t+1}(X_{t+1}^{(i)} \mid \boldsymbol{X}_t^{(i)}) g_{t+1}(Y_{t+1} \mid X_{t+1}^{(i)})}{q_{t+1}(X_{t+1}^{(i)})}$$

Sequential Importance Sampling (SIS)

1. **Initialization:**
   1.1 Generate N independent samples $X_0^{(i)}$ from the proposal distribution $q_0(X_0)$.
   1.2 Assign weights $w_0^{(i)} \propto f_0(X_0^{(i)})/q_0(X_0^{(i)})$.
2. **Iteration:** For $t = 1, 2, \ldots, T$,
   2.1 Sample $X_t^{(i)} \sim q_t(X_t)$ for $i = 1, \ldots, N$.
   2.2 Assign weights

$$w_t^{(i)} \propto w_{t-1}^{(i)} \frac{f_t(X_t^{(i)} \mid \boldsymbol{X}_{t-1}^{(i)}) g_t(Y_t \mid X_t^{(i)})}{q_t(X_t^{(i)})}$$

Then:

- The weighted samples $(\boldsymbol{X}_t^{(i)}, w_t^{(i)})_{i=1}^N$ are samples from the filtering distribution $p(\boldsymbol{X}_t \mid \boldsymbol{Y}_t)$.
- The weighted samples $(\boldsymbol{X}_T^{(i)}, w_T^{(i)})_{i=1}^N$ are samples from the smoothing distribution $p(\boldsymbol{X}_T \mid \boldsymbol{Y}_{1:T})$.

Justification on the Importance Sampling

- From the principle of impoartance sampling, if $X^{(i)}$ are samples from $q(X)$ and $(X^{(i)}, w^{(i)})$ are (weighted) samples from the target $p(X)$, then

$$w^{(i)} \propto \frac{p(X^{(i)})}{q(X^{(i)})}$$

- For the SIS algorithm, the sampling distributions for $\boldsymbol{X}_t$ is

$$q(\boldsymbol{X}_t) = q_0(X_0) \prod_{s=1}^{t} q_s(X_s)$$

- The target filtering distribution is

$$p(\boldsymbol{X}_t \mid \boldsymbol{Y}_t) \propto f_0(X_0) \prod_{s=1}^{t} f_s(X_s \mid \boldsymbol{X}_{s-1}) g_s(Y_s \mid X_s)$$

Justification on the Importance Sampling

- The proper weight for the i-th sample at time t is

$$w_t^{(i)} \propto \frac{p(\boldsymbol{X}_t^{(i)} \mid \boldsymbol{Y}_t)}{q(\boldsymbol{X}_t^{(i)})} \propto \frac{q_0(X_0^{(i)})}{f_0(X_0^{(i)})} \prod_{s=1}^{t} \frac{f_s(X_s^{(i)} \mid \boldsymbol{X}_{s-1}^{(i)}) g_s(Y_s \mid X_s^{(i)})}{q_s(X_s^{(i)})}$$

- On the one hand, this is the cumulated product of the importance weights for the samples up to time t:

$$w_t^{(i)} \propto w_0^{(i)} \prod_{s=1}^{t} \frac{w_s^{(i)}}{w_{s-1}^{(i)}}$$

- On the other hand, the sequential update for the weights is

$$w_t^{(i)} \propto w_{t-1}^{(i)} \frac{f_t(X_t^{(i)} \mid \boldsymbol{X}_{t-1}^{(i)}) g_t(Y_t \mid X_t^{(i)})}{q_t(X_t^{(i)})}$$

Different Choices for the Proposal Distribution

- Particle Filter / Bootstrap Filter:

$$q_t(X_t) = f_t(X_t \mid \boldsymbol{X}_{t-1})$$

- Independent Filter:

$$q_t(X_t) \propto g_t(Y_t \mid X_t)$$

- Conditional Optimal Filter:

$$q_t(X_t) \propto f_t(X_t \mid \boldsymbol{X}_{t-1}) g_t(Y_t \mid X_t)$$

- Auxiliary Particle Filter:

$$q_t(X_t) \propto p(Y_{t+1} \mid X_t)$$

Likelihood Estimation with SIS

Suppose the state-space model dynamics is parametrized by $\boldsymbol{\theta}$ and we want to estimate the likelihood $p(\boldsymbol{Y}_{1:T} \mid \boldsymbol{\theta})$.

- The likelihood can be written as a high-dimensional integral:

$$
\begin{aligned}
p(\boldsymbol{Y}_T \mid \boldsymbol{\theta}) &= \int p(\boldsymbol{Y}_T, \boldsymbol{X}_T \mid \boldsymbol{\theta}) d\boldsymbol{X}_T \\
&= \int f_0(X_0 \mid \boldsymbol{\theta}) \prod_{s=1}^{T} f_s(X_s \mid \boldsymbol{X}_{s-1}; \boldsymbol{\theta}) g_s(Y_s \mid X_s; \boldsymbol{\theta}) d\boldsymbol{X}_T
\end{aligned}
$$

- Directly estimate the likelihood is infeasible due to the high-dimensional integral.

Likelihood Estimation with SIS

With SIS, we observe that

$$\begin{aligned}
\mathbb{E}_{\text{SIS}}\left[\frac{w_t}{w_{t-1}}\right] &= \mathbb{E}_{\text{SIS}}\left[\frac{f_t(X_t \mid \boldsymbol{X}_{t-1};\boldsymbol{\theta})g_t(Y_t \mid X_t;\boldsymbol{\theta})}{q_t(X_t)}\right] \\
&= \int \frac{f_t(X_t \mid \boldsymbol{X}_{t-1};\boldsymbol{\theta})g_t(Y_t \mid X_t;\boldsymbol{\theta})}{q_t(X_t)} q_t(X_t) p(\boldsymbol{X}_{t-1} \mid \boldsymbol{Y}_{t-1};\boldsymbol{\theta}) dX_t d\boldsymbol{X}_{t-1} \\
&= \int f_t(X_t \mid \boldsymbol{X}_{t-1};\boldsymbol{\theta})g_t(Y_t \mid X_t;\boldsymbol{\theta}) p(\boldsymbol{X}_{t-1} \mid \boldsymbol{Y}_{t-1};\boldsymbol{\theta}) dX_t d\boldsymbol{X}_{t-1} \\
&= \int \left(\int f_t(X_t \mid \boldsymbol{X}_{t-1};\boldsymbol{\theta}) p(\boldsymbol{X}_{t-1} \mid \boldsymbol{Y}_{t-1};\boldsymbol{\theta}) d\boldsymbol{X}_{t-1}\right) g_t(Y_t \mid X_t;\boldsymbol{\theta}) dX_t \\
&= \int p(X_t \mid \boldsymbol{Y}_{t-1};\boldsymbol{\theta}) g_t(Y_t \mid X_t;\boldsymbol{\theta}) dX_t \\
&= p(Y_t \mid \boldsymbol{Y}_{t-1};\boldsymbol{\theta})
\end{aligned}$$

Likelihood Estimation with SIS

Notice that

$$p(\boldsymbol{Y}_t; \boldsymbol{\theta}) = \prod_{s=1}^{T} p(Y_t \mid \boldsymbol{Y}_{t-1}; \boldsymbol{\theta})$$

1. **Initialization:**
   1.1 Set $L = 1$.
   1.2 Generate N independent samples $X_0^{(i)}$ from the proposal distribution $q_0(X_0)$.
   1.3 Assign weights $w_0^{(i)} \propto f_0(X_0^{(i)})/q_0(X_0^{(i)})$.
2. **Iteration:** For $t = 1, 2, \ldots, T$,
   2.1 Sample $X_t^{(i)} \sim q_t(X_t)$ for $i = 1, \ldots, N$.
   2.2 Assign weights

$$w_t^{(i)} = w_{t-1}^{(i)} \frac{f_t(X_t^{(i)} \mid \boldsymbol{X}_{t-1}^{(i)}) g_t(Y_t \mid X_t^{(i)})}{q_t(X_t^{(i)})}$$

   2.3 Update the likelihood estimate

$$L = L \cdot \frac{\sum_{i=1}^{N} w_t^{(i)}}{\sum_{i=1}^{N} w_{t-1}^{(i)}}$$

Example

Consider a simple state-space model with the following dynamics:

$$X_t \mid X_{t-1} \sim \mathcal{N}(\phi X_{t-1}, 1)$$
$$Y_t \mid X_t \sim \mathcal{N}(X_t, 1)$$

where ϕ is the parameter to be estimated.

Example

Simulate data from the model with $\phi = 0.6$.

```
T = 20
Y = rep(0, T)
X = 0
for(t in 1:T){
    X = 0.6 * X + rnorm(1)
    Y[t] = X + rnorm(1)
}
```

Example

Compute the likelihood with SIS:

```
llh <- function(phi){
    n = 1000
    x = rep(0, n)
    logw = rep(0, n)
    loglik = 0
    for(t in 1:T){
        z = rnorm(n)/sqrt(2)
        xx = (phi*x + Y[t])/2 + z
        dlogw = -0.5*(xx - phi*x)**2
        dlogw = dlogw - 0.5*(Y[t]-xx)**2
        dlogw = dlogw + z**2
        x = xx
        loglik = loglik + log(sum(exp(logw+dlogw)))
        loglik = loglik - log(sum(exp(logw)))
        logw = logw + dlogw
        logw  = logw - mean(logw)
    }
    return(loglik)
}
```

Example

Compute the MLE:

```
phi.hat = optimize(llh, c(-1, 1), maximum = T)$maximum
```

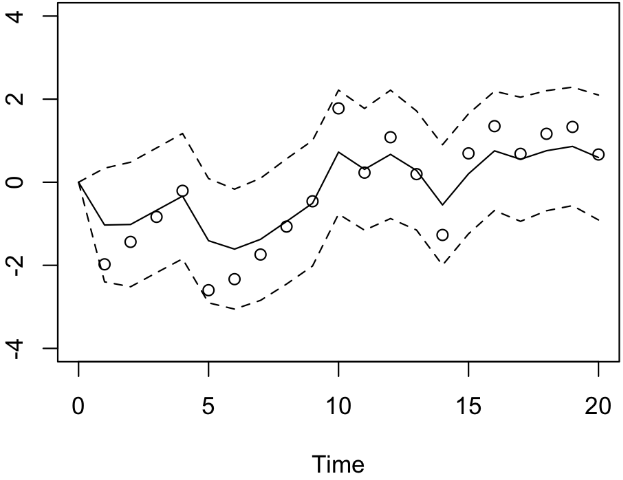
The outcome is $\hat{\phi} = 0.61$. (The result can be noisy due to the randomness in the SIS algorithm and lack of resampling.)

Example

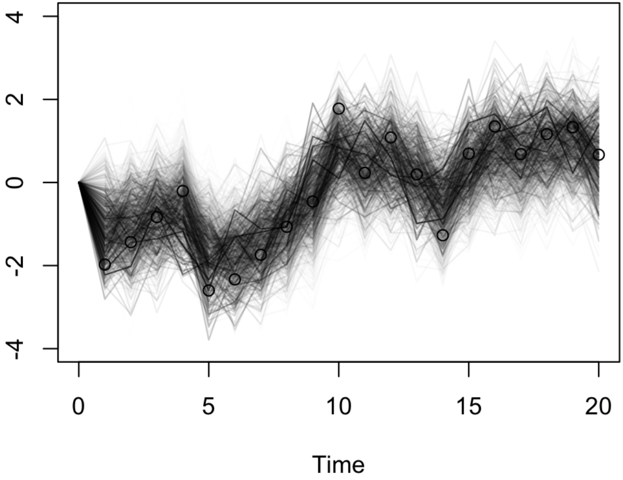
Draw samples from the posterios:

```
smc <- function(phi){
    n = 1000
    x = array(0, c(n, T+1))
    logw = rep(0, n)
    for(t in 1:T){
        z = rnorm(n)/sqrt(2)
        x[,t+1] = (phi*x[,t] + Y[t])/2 + z
        dlogw = -0.5*(x[,t+1] - phi*x[,t])**2
        dlogw = dlogw - 0.5*(Y[t]-x[,t+1])**2
        dlogw = dlogw + z**2
        logw = logw + dlogw
        logw  = logw - mean(logw)
    }
    return(x)
}
```

Example

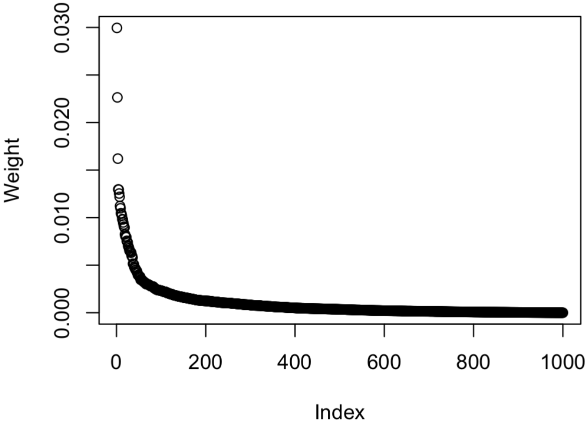


Example



Degeneracy

One of the problem is the degeneracy of the SIS algorithm. The weights of the particles can be very skewed, leading to poor performance of the algorithm.



Degeneracy

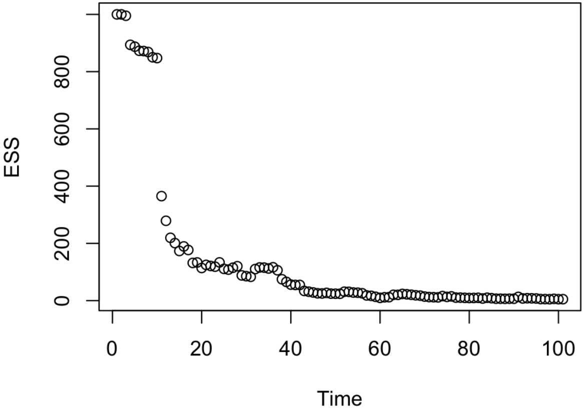
One way to evaluate the performance of the SIS algorithm is to look at the effective sample size (ESS):

$$\text{ESS}_t = \frac{\left(\sum_{i=1}^{N} w_t^{(i)}\right)^2}{\sum_{i=1}^{N} (w_t^{(i)})^2}$$

- The ESS of the previous example at time T is $\sim 183 \ll 1000$.
- If the observation equation is restrictive, the weight adjustedment step can lead to a large variance in the weights, resulting in a low ESS.
- We refer to the problem of **reduced effective sample size** as **degeneracy**.

Example

For the previous Autoregressive example, if we set $T = 100$, the ESS is tracked over time as follows.



Resampling

One way to alleviate the degeneracy problem is to introduce **resampling** steps in the SIS algorithm.

- Suppose now we have $N = 5$ samples at time t:

$$(X_t^{(1)}, 0.8),\ (X_t^{(2)}, 0.17),\ (X_t^{(3)}, 0.01),\ (X_t^{(4)}, 0.01),\ (X_t^{(5)}, 0.01),$$

  where the second element is the weight.

- **Without resampling**, the samples at time $t+1$ will be dominated by the first sample:

$$(X_{t+1}^{(1)}, 0.83),\ (X_{t+1}^{(2)}, 0.14),\ (X_{t+1}^{(3)}, 0.01),\ (X_{t+1}^{(4)}, 0.01),\ (X_{t+1}^{(5)}, 0.01)$$

- **With resampling**, we draw $N = 5$ samples from the current samples with replacement:

$$(X_t^{(1)}, 0.2),\ (X_t^{(1)}, 0.2),\ (X_t^{(1)}, 0.2),\ (X_t^{(1)}, 0.2),\ (X_t^{(2)}, 0.2),$$

SIS with Resampling (SISR)

1. **Initialization:**
   1.1 Generate N independent samples $X_0^{(i)}$ from the proposal distribution $q_0(X_0)$.
   1.2 Assign weights $w_0^{(i)} \propto f_0(X_0^{(i)})/q_0(X_0^{(i)})$.
2. **Iteration:** For $t = 1, 2, \ldots, T$,
   2.1 Sample $X_t^{(i)} \sim q_t(X_t)$ for $i = 1, \ldots, N$.
   2.2 Assign weights

$$w_t^{(i)} = w_{t-1}^{(i)} \frac{f_t(X_t^{(i)} \mid \boldsymbol{X}_{t-1}^{(i)}) g_t(Y_t \mid X_t^{(i)})}{q_t(X_t^{(i)})}$$

   2.3 (Optional) Resample N samples from $\{X_t^{(i)}\}_{i=1}^N$ with replacement according to the weights $\{w_t^{(i)}\}_{i=1}^N$ and set weights to be $\propto 1$.

When to Resample?

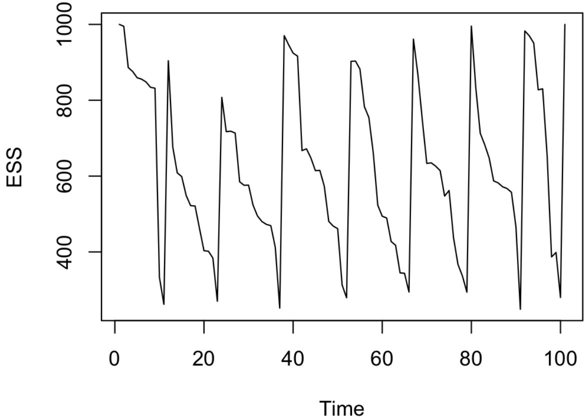
- Resampling is a trade-off between the variance reduction and the information loss.
- If the weights are very skewed, resampling can help to reduce the variance of the weights.
- If the weights are not very skewed, resampling can lead to information loss.

Resampling schedules:

- Deterministic schedule: Resample every K steps.
- Dynamic schedule: Resample when the ESS is below a threshold.

Example

For the previous Autoregressive example, if we set resample when ESS is below $0.3N$, the ESS is tracked over time as follows.



Resampling w.r.t. the Priority Scores

- The resampling step can be modified to incorporate the priority scores.
- The priority scores are the weights of the samples in the resampling step.
- The resampling step w.r.t. the priority scores β_i is:
  1. Draw N samples $\{j_1, \ldots, j_N\}$ with replacement from $\{1, \ldots, N\}$ with probabilities (proportional to) $\{\beta_i\}_{i=1}^N$.
  2. Set the new samples to be $\{X_t^{(j_1)}, \ldots, X_t^{(j_N)}\}$.
  3. Set the new weights to be

$$w^{(j_i)} \leftarrow \frac{w^{(j_i)}}{\beta_{j_i}}$$

- The previous example is a special case with $\beta_i = w_t^{(i)}$.
- **Least Aggresive Resampling:** Set $\beta_i = \sqrt{w_t^{(i)}}$ for all i.

How to Resample?

- The resampling step can be implemented in different ways.
- **Simple Random Resampling**: Draw N samples with replacement from $\{1, \dots, N\}$ with probabilities $\{\beta_i\}_{i=1}^N$.
- **Residual Resampling**:
  1. Retain $k_i = \lfloor N\tilde{w}^{(i)}$ copies of $X^{(i)}$, where $\tilde{w}^{(i)} = w^{(i)} / \sum_i w^{(i)}$.
  2. Obtain $N - \sum_i k_i$ samples by drawing with replacement from $\{1, \dots, N\}$ with probabilities $N\tilde{w}^{(i)} - k_i$.

Sequential Importance Sampling with Resampling (SISR)

1. **Initialization:**
   1.1 Generate N independent samples $X_0^{(i)}$ from the proposal distribution $q_0(X_0)$.
   1.2 Assign weights $w_0^{(i)} \propto f_0(X_0^{(i)})/q_0(X_0^{(i)})$.
2. **Iteration:** For $t = 1, 2, \ldots, T$,
   2.1 Sample $X_t^{(i)} \sim q_t(X_t)$ for $i = 1, \ldots, N$.
   2.2 Assign weights
   $$w_t^{(i)} = w_{t-1}^{(i)} \frac{f_t(X_t^{(i)} \mid \boldsymbol{X}_{t-1}^{(i)}) g_t(Y_t \mid X_t^{(i)})}{q_t(X_t^{(i)})}$$
   2.3 Conduct computation w.r.t. to the filtering sample here.
   2.4 (Optional Resampling):
      2.4.1 Draw N samples with replacement from $\{1, \ldots, N\}$ with probabilities $\{\beta_i\}_{i=1}^N$.
      2.4.2 Set the new samples to be $\{\boldsymbol{X}_t^{(j_1)}, \ldots, \boldsymbol{X}_t^{(j_N)}\}$.
      2.4.3 Set the new weights to be
      $$w^{(j_i)} \leftarrow \frac{w^{(j_i)}}{\beta_{j_i}}$$
3. Conduct computation w.r.t. to the smoothing sample here.

Example

We consider the following 1D random walk with noisy observations:

$$X_t = X_{t-1} + \mathcal{N}(0, 1)$$
$$Y_t = X_t + \mathcal{N}(0, 1)$$

The starting point is $X_0 = 0$.

Example

Simulate data from the model:

```
T = 100
x = cumsum(rnorm(T))
y = x + rnorm(T)
```

Example

```
smc <- function(n, y, resample=FALSE){
    T = length(y)
    X = array(0, dim=c(n, T+1))
    logw = rep(0, n)
    out.filter = rep(0, T)
    ess = rep(n, T)
    for(t in 1:T){
        z = rnorm(n) / sqrt(2)
        X[,t+1] = (X[,t] + y[t]) / 2 + z
        logw = logw -0.5*(y[t]-X[,t+1])**2 - 0.5*(X[,t+1]-X[,t])
            **2
        logw = logw + 0.5*z**2
        logw = logw - mean(logw)
        w = exp(logw)
        w = w / sum(w)
        out.filter[t] = X[,t+1]%*%w
        ess[t] = sum(w)**2/sum(w**2)
```

Example

```
        if(resample && ess[t] < 0.3*n){
            index = sample(n, n, replace=T, prob=w)
            logw = rep(0, n)
            X = X[index,]
        }
    }
    w = exp(logw)
    w = w / sum(w)
    out.smoothing = w%*%X
    return(list(filtering=out.filter, smoothing=out.smoothing,
        ess=ess))
}
```