

# Particle Filtering

STATS 744

Oct. 22nd, 2015

# Outline

- 1 Introduction
- 2 Preliminary
  - Importance Sampling
  - Hidden Markov Model
- 3 The Basic Particle Filtering
- 4 Numerical Example
- 5 Iterative Filtering

# Outline

## 1 Introduction

## 2 Preliminary

- Importance Sampling
- Hidden Markov Model

## 3 The Basic Particle Filtering

## 4 Numerical Example

## 5 Iterative Filtering

# Introduction

- Usually, prior knowledge about the phenomenon being modelled is available.
- Inferences on unknown quantities are based on the posterior.
- It is necessary to update the posterior distribution when new data is available.
- New observations come sequentially.
- For example, tracking an aircraft using radar, estimating the volatility of financial instruments using stock market data.

- If the data are modelled by a linear Gaussian state-space model, we can derive an exact analytical expression to compute the evolving sequence of posterior distribution by Kalman filter.
- If the data are modelled as partially observed, finite state-space Markov chain, we can obtain the analytical solution by hidden Markov model.
- Sequential Monte Carlo methods are a set of simulation-based methods which provide a convenient and attractive approach to computing the posterior distribution.

# Outline

## 1 Introduction

## 2 Preliminary

- Importance Sampling
- Hidden Markov Model

## 3 The Basic Particle Filtering

## 4 Numerical Example

## 5 Iterative Filtering

# Importance Sampling

- Suppose one is interested in evaluating the expected value

$$E_{\pi}(f(X)) = \int f(x)\pi(x)dx.$$

- Sampling from  $\pi$  is hard but we can easily sample from  $g$ .
- If  $g$  is an *importance density* having property that  $g(x) = 0$  implies  $\pi(x) = 0$ , then one can write

$$E_{\pi}(f(X)) = \int f(x) \frac{\pi(x)}{g(x)} g(x) dx = E_g(f(X)\omega(X)).$$

where  $\omega(x) = \frac{\pi(x)}{g(x)}$  is *importance function*.

- Approximate the expected value of interest by generating a random sample of size  $N$  from  $g$  and computing

$$\frac{1}{N} \sum_{i=1}^N f(x^{(i)})\omega(x^{(i)}) \approx E_{\pi}(f(X)).$$

# Hidden Markov Model

- For unobserved true discrete-state Markov process  $X_t$ , it can only be inaccurately measured through a variable  $Y_t$ .
- Assume an initial distribution for  $X_0$ ;
- The true process evolves to the next time step

$$X_{t+1} \sim f_1(X_t, \theta);$$

- The observed process for each time point

$$Y_t \sim f_2(X_t, \theta),$$

where  $f_i$  are some known distributions and  $\theta$  is a parameter vector.



- From the observations  $Y_{1:t}$ , we can sample  $X_t$  from the target posterior distribution  $p(X_t|Y_{1:t})$  by using

$$\begin{aligned} p(X_t|Y_{1:t}) &= \frac{p(X_t, Y_{1:t})}{p(Y_{1:t})} \\ &= \frac{p(Y_t|X_t)p(X_t|Y_{1:t-1})}{p(Y_{1:t})} \\ &\propto p(Y_t|X_t)p(X_t|Y_{1:t-1}) \end{aligned}$$

- We can sample from importance density  $p(X_t|Y_{1:t-1})$  by  $f_1$  and  $f_2$ .
- $p(Y_t|X_t)$  is importance function.

# Outline

- 1 Introduction
- 2 Preliminary
  - Importance Sampling
  - Hidden Markov Model
- 3 The Basic Particle Filtering**
- 4 Numerical Example
- 5 Iterative Filtering

# The Basic Particle Filtering

- Particle filtering is how sequential Monte Carlo is usually referred to in applications to state space models.
- This method is easier to understand when viewed as an extension of importance sampling.
- Give a starting particles based on the distribution assumption for  $X_0$ .
- Resample  $X_0$  with replacement according to weights.
- The resampled particles are used to predict  $X_1$ .
- Take  $X_1$  as starting point for the next iteration.
- The algorithm can be described as next page.

**Data:** Observation time series  $y_1, \dots, y_T$

**Input:** number of particles  $N$ , initial sampling distribution  $p_0$

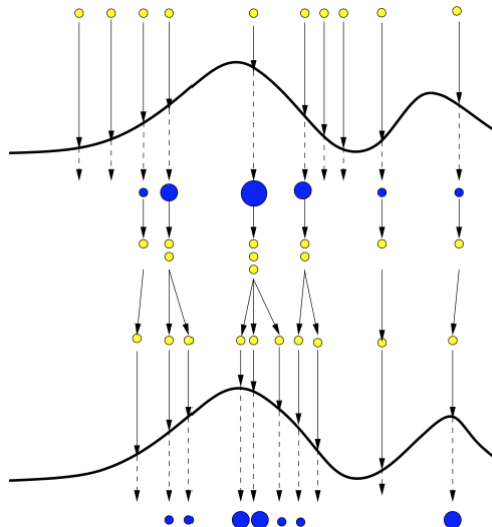
```

/* Initialization */
1 Sample  $N$  particles from  $p_0$ :  $x^1, \dots, x^N$ 
2 for ( $t = 1, 2, \dots, T$ ) do
    /* Weights from likelihood (measurement update) */
    3 for ( $i = 1, 2, \dots, N$ ) do
    4      $w^i \leftarrow p(y_t | x^i) = f_2(x^i, \theta)$ 
    5 end
    /* Normalize weights */
    6 for ( $i = 1, 2, \dots, N$ ) do
    7      $w^i \leftarrow w^i / \sum_k w^k$ 
    8 end
    /* Resample according to weights */
    9  $\tilde{x}^{1:N} \leftarrow \text{sample}(x^{1:N}, \text{prob} = w, \text{replace} = \text{true})$ 
    /* Update next time step  $X$  from the resampled particles: predicts the new particles */
    10 for ( $i = 1, 2, \dots, N$ ) do
    11      $x^i \leftarrow f_1(\tilde{x}^i, \theta)$ 
    12 end
13 end

```

**Result:** A set of  $N$  samples approximating the continuous posterior distribution  $p(x_t | y_t)$  at every time step  $t = 1, \dots, T$ .

**Figure.** Pseudocode by David Champredon



**Figure.** Design by David Champredon

# Outline

- 1 Introduction
- 2 Preliminary
  - Importance Sampling
  - Hidden Markov Model
- 3 The Basic Particle Filtering
- 4 Numerical Example
- 5 Iterative Filtering

## Numerical Example

- The data are generated from a local level model with system variance 1 and observation variance 2.
- The initial distribution is  $N(10, 9)$ .
- We record the observations  $Y$ .
- The number of particles is 1000.
- Setting the threshold step to 500 then we do resampling whenever the effective sample size drops below one half of the number of particles.
- We can compare the filtering state estimates and their deviations computed with the Kalman filter and particle filter.

## Part 1: initialization

```
library("dlm")

### Generate data
mod <- dlmModPoly(1,dV=2,dW=1,m0=10,C0=9)
n <- 100
set.seed(23)
simData <- dlmForecast(mod=mod,nAhead=n,sampleNew=1)
y <- simData$newObs[[1]]

### Basic Particle Filter - optimal importance density
N <- 1000
N_0 <- N/2
pfOut <- matrix(NA_real_,n+1,N)
wt <- matrix(NA_real_,n+1,N)
importanceSd <- sqrt(drop(W(mod)-W(mod)^2/(W(mod)+V(mod))))
predSd <- sqrt(drop(W(mod)+V(mod)))
```



## Part 2: sampling

```

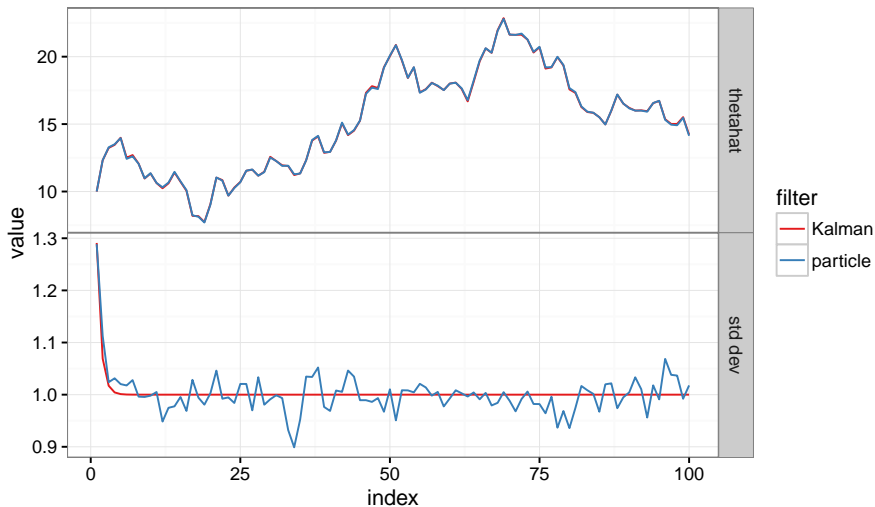
pfOut[1,] <- rnorm(N,mean=m0(mod),sd=sqrt(C0(mod))); wt[1,] <- rep(1/N,N)
for (it in 2:(n+1)) {
  ## generate particles
  means <- pfOut[it-1,]+W(mod)*(y[it-1]-pfOut[it-1,])/(W(mod)+V(mod))
  pfOut[it,] <- rnorm(N,mean=means,sd=importanceSd)
  ## update the weights
  wt[it,] <- dnorm(y[it-1],mean=pfOut[it-1,],sd=predSd)*wt[it-1,]
  wt[it,] <- wt[it,]/sum(wt[it,])
  N.eff <- 1/crossprod(wt[it,]) ## need to resample?
  if (N.eff < N_0) { ## multinomial resampling
    index <- sample(N,N,replace=TRUE,prob=wt[it,])
    pfOut[it,] <- pfOut[it,index]
    wt[it,] <- 1/N
  }
}

```

## Part 3

```
### Compare exact filtering distribution with PF approximation
modFilt <- dlmFilter(y,mod)
thetaHatKF <- modFilt$m[-1]
sdKF <- with(modFilt,sqrt(unlist(dlmSvd2var(U.C,D.C))))[-1]
pfOut <- pfOut[-1,]
wt <- wt[-1,]
thetaHatPF <- sapply(1:n,function(i) weighted.mean(pfOut[i,],wt[i,]))
sdPF <- sapply(1:n,function(i)
  sqrt(weighted.mean((pfOut[i,]-thetaHatPF[i])^2,wt[i,])))
```

# Particle vs Kalman filter results

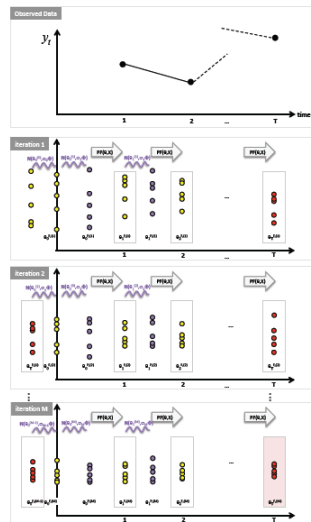


# Outline

- 1 Introduction
- 2 Preliminary
  - Importance Sampling
  - Hidden Markov Model
- 3 The Basic Particle Filtering
- 4 Numerical Example
- 5 Iterative Filtering**

# Iterative Filtering

- Instead of performing a single particle filter from the observed data  $Y_{1:T}$ , we estimate the posterior distribution of  $X_t$  at each time step.
- This method iterates several times through the process.
- The idea is changing the constant parameter  $\theta$  into the dynamic  $\theta(t)$  following a Gaussian process.
- It converges to maximum likelihood estimator since the variance of  $\theta(t)$  tends to 0.
- The algorithm is listed on next page.



**Figure.** Design by David Champredon

**Data:** Observation time series  $y_1, \dots, y_T$

**Input:** number of main iterations  $M$ , number of particles  $J$ ,  $J$  initial parameter particles  $\Theta_j^{(0)}$ , perturbation density  $h(\text{mean}, \text{var})$ , perturbation sequence matrix  $\sigma_{1T}$ , function  $f$  evaluating the process  $X$  at the next time step, observation process  $g(y|X, \theta)$

```

1 for  $m = 1, 2, \dots, M$  do
2   for  $j = 1, \dots, J$  do
3     /* Process starting position */
4      $\theta_F^{(m)}(0, j) \sim h(\Theta^{(m-1)}_j, \sigma_m)$ 
5      $X_F(0, j) = f(X(0), \theta_F^{(m)}(0, j))$ 
6   end
7   /* Loop through all observation dates */
8   for  $t = 1, 2, \dots, T$  do
9     /* Particle filter from t-1 to t: */
10    for  $j = 1, \dots, J$  do
11      /* Perturbation of parameters: */
12       $\theta_P^{(m)}(t, j) \sim h(\theta_F^{(m)}(t-1, j), \sigma_m)$ 
13      /* Prediction based on perturbation: */
14       $X_P(t, j) \leftarrow f(X_F(t-1, j), \theta_P^{(m)}(t, j))$ 
15      /* weights from perturbed likelihood */
16       $w(t, j) \leftarrow g(y_t | X_P(t, j), \theta_P^{(m)}(t, j))$ 
17    end
18     $k_1, \dots, k_J \leftarrow \text{sample}(1 : J, \text{prob} = w(t, \bullet), \text{replace} = \text{true})$ 
19    /* Update parameter and process */
20    for  $j = 1, \dots, J$  do
21       $\theta_F^{(m)}(t, j) \leftarrow \theta_P^{(m)}(t, k_j)$ 
22       $X_F(t, j) \leftarrow X_P(t, k_j)$ 
23    end
24    /* ((End particle filter)) */
25  end
26  /* Set all particles for next main iteration */
27   $\Theta^{(m)}_j \leftarrow \theta_F^{(m)}(t, j)$  for all  $j = 1, \dots, J$ 
28 end

```

**Figure.** Pseudocode by David Champredon

# Thanks For Your Patience!