Time Series and Sequence Learning

Lecture 8 – Parameter Estimation in Non-Linear/Non-Gaussian State Space Models

Johan Alenlöv, Linköping University

2024-09-30

Summary of Lecture 7

General State-Space Models

Def. A General State-Space model is given by:

$$\alpha_t \mid \alpha_{t-1} \sim q(\alpha_t \mid \alpha_{t-1})$$
$$y_t \mid \alpha_t \sim g(y_t \mid \alpha_t)$$

and initial distribution $\alpha_1 \sim q(\alpha_1)$.

General State-Space Models

Def. A General State-Space model is given by:

$$\alpha_t \mid \alpha_{t-1} \sim q(\alpha_t \mid \alpha_{t-1})$$
$$y_t \mid \alpha_t \sim g(y_t \mid \alpha_t)$$

and initial distribution $\alpha_1 \sim q(\alpha_1)$.

Thm. The joint-smoothing distribution is given by

$$p(\alpha_{1:n} \mid y_{1:n}) = \frac{q(\alpha_1)g(y_1 \mid \alpha_1)\prod_{i=2}^{n} q(\alpha_i \mid \alpha_{i-1})g(y_i \mid \alpha_i)}{L_n(y_{1:n})},$$

where $L_n(y_{1:n}) = \int q(\alpha_1)g(y_1 \mid \alpha_1)\prod_{i=2}^{n} q(\alpha_i \mid \alpha_{i-1})g(y_i \mid \alpha_i)\mathrm{d}\alpha_{1:n}$ is the likelihood.

Sequential Importance Sampling

- A first idea was to use importance sampling targeting the joint-smoothing distribution.

Sequential Importance Sampling

- A first idea was to use importance sampling targeting the joint-smoothing distribution.
- We go from time n to $n+1$ in the following way:
  - Draw $\alpha^i_{n+1} \sim f(\alpha_{n+1} \mid \alpha^i_{1:n})$ (propagating)
  - Set $\alpha^i_{1:n+1} = (\alpha^i_{1:n}, \alpha^i_{n+1})$
  - Set $\omega^i_{n+1} = \frac{q(\alpha^i_{n+1} \mid \alpha^i_n) g(y_{n+1} \mid \alpha^i_{n+1})}{f(\alpha^i_{n+1} \mid \alpha^i_{1:n})} \times \omega^i_n$. (weighting)

Sequential Importance Sampling

- A first idea was to use importance sampling targeting the joint-smoothing distribution.
- We go from time n to $n+1$ in the following way:
  - Draw $\alpha_{n+1}^i \sim f(\alpha_{n+1} \mid \alpha_{1:n}^i)$ (propagating)
  - Set $\alpha_{1:n+1}^i = (\alpha_{1:n}^i, \alpha_{n+1}^i)$
  - Set $\omega_{n+1}^i = \frac{q(\alpha_{n+1}^i \mid \alpha_n^i) g(y_{n+1} \mid \alpha_{n+1}^i)}{f(\alpha_{n+1}^i \mid \alpha_{1:n}^i)} \times \omega_n^i$. (weighting)
- Simplest choice: $f(\alpha_{n+1} \mid \alpha_{1:n}^i) = q(\alpha_{n+1} \mid \alpha_n^i)$

Sequential Importance Sampling

- A first idea was to use importance sampling targeting the joint-smoothing distribution.
- We go from time n to $n+1$ in the following way:
  - Draw $\alpha^i_{n+1} \sim f(\alpha_{n+1} \mid \alpha^i_{1:n})$ (propagating)
  - Set $\alpha^i_{1:n+1} = (\alpha^i_{1:n}, \alpha^i_{n+1})$
  - Set $\omega^i_{n+1} = \frac{q(\alpha^i_{n+1} \mid \alpha^i_n) g(y_{n+1} \mid \alpha^i_{n+1})}{f(\alpha^i_{n+1} \mid \alpha^i_{1:n})} \times \omega^i_n$. (weighting)
- Simplest choice: $f(\alpha_{n+1} \mid \alpha^i_{1:n}) = q(\alpha_{n+1} \mid \alpha^i_n)$
- We estimate using:

$$\sum_{i=1}^{N} \frac{\omega^i_{n+1}}{\Omega_{n+1}} h(\alpha^i_{n+1}) \approx \mathbb{E}[h(\alpha_{n+1}) \mid y_{1:n+1}]$$

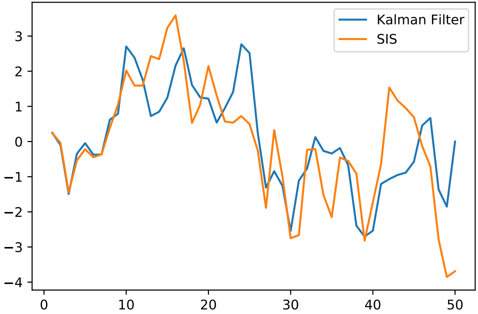
$$N^{-1}\Omega_{n+1} = \frac{1}{N} \sum_{i=1}^{N} \omega^i_{n+1} \approx L(y_{1:n+1})$$

Example: Linear Gaussian State Space Model

Def. A Linear Gaussian State-Space (LGSS) model is given by:

$$\alpha_t = T\alpha_{t-1} + R\eta_t, \qquad \eta_t \sim \mathcal{N}(0, Q),$$
$$y_t = Z\alpha_t + \varepsilon_t \qquad \varepsilon_t \sim \mathcal{N}(0, \sigma_\epsilon^2),$$

and initial distribution $\alpha_1 \sim \mathcal{N}(a_1, P_1)$.

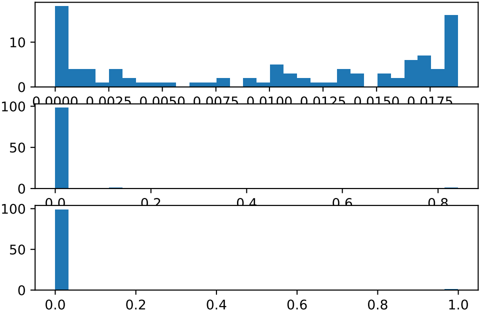


Example: Linear Gaussian State Space Model

Def. A Linear Gaussian State-Space (LGSS) model is given by:

$$\alpha_t = T\alpha_{t-1} + R\eta_t, \qquad \eta_t \sim \mathcal{N}(0, Q),$$
$$y_t = Z\alpha_t + \varepsilon_t \qquad \varepsilon_t \sim \mathcal{N}(0, \sigma_\epsilon^2),$$

and initial distribution $\alpha_1 \sim \mathcal{N}(a_1, P_1)$.



Sequential Importance Sampling with Resampling

- We solve the problem of weight degeneracy by resampling the particles!

Sequential Importance Sampling with Resampling

- We solve the problem of weight degeneracy by resampling the particles!
- The most natural selection is to draw new particles $(\tilde{\alpha}^i_{1:n})^N_{i=1}$ among the SIS produced $(\alpha^i_{1:n})^N_{i=1}$ with probabilities given by the normalized importance weights.

Sequential Importance Sampling with Resampling

- We solve the problem of weight degeneracy by resampling the particles!
- The most natural selection is to draw new particles $(\tilde{\alpha}^i_{1:n})_{i=1}^N$ among the SIS produced $(\alpha^i_{1:n})_{i=1}^N$ with probabilities given by the normalized importance weights.
- For $i = 1, 2, \ldots, N$ we let

$$\tilde{\alpha}^i_{1:n} = \alpha^j_{1:n} \quad \text{w. pr.} \quad \frac{\omega^j_n}{\Omega_n}$$

Sequential Importance Sampling with Resampling

- We solve the problem of weight degeneracy by resampling the particles!
- The most natural selection is to draw new particles $(\tilde{\alpha}^i_{1:n})^N_{i=1}$ among the SIS produced $(\alpha^i_{1:n})^N_{i=1}$ with probabilities given by the normalized importance weights.
- For $i = 1, 2, \ldots, N$ we let

$$\tilde{\alpha}^i_{1:n} = \alpha^j_{1:n} \quad \text{w. pr.} \quad \frac{\omega^j_n}{\Omega_n}$$

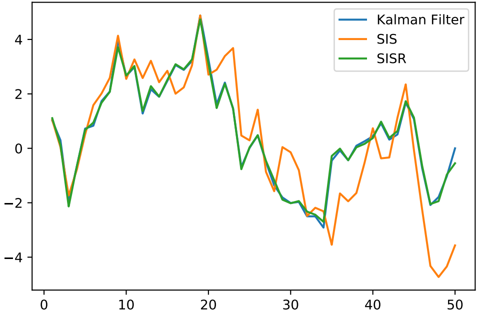
- Add this step before propagating the particles!

Example: Linear Gaussian State Space Model

Def. A Linear Gaussian State-Space (LGSS) model is given by:

$$\alpha_t = T\alpha_{t-1} + R\eta_t, \qquad \eta_t \sim \mathcal{N}(0, Q),$$
$$y_t = Z\alpha_t + \varepsilon_t \qquad \varepsilon_t \sim \mathcal{N}(0, \sigma_\epsilon^2),$$

and initial distribution $\alpha_1 \sim \mathcal{N}(a_1, P_1)$.



Algorithm: Particle Filter

Particle Filter:

Draw $\alpha_1^i \sim f(\alpha_1)$
Set $\omega_1^i = \frac{q(\alpha_1^i)g(y_1 \mid \alpha_1^i)}{f(\alpha_1^i)}$
Set $\Omega_1 = \sum_{i=1}^N \omega_1^i$
for $t = 2, 3, \ldots, n$ **do**
  Draw $I^i = j$ w. pr. $\frac{\omega_{t-1}^j}{\Omega_{t-1}}$
  Draw $\alpha_t^i \sim f(\alpha_t \mid \alpha_{1:t-1}^{I^i})$
  Set $\omega_t^i = \frac{q(\alpha_t^i \mid \alpha_{t-1}^{I^i})g(y_t \mid \alpha_t^i)}{f(\alpha_t^i \mid \alpha_{1:t-1}^{I^i})}$
  Set $\alpha_{1:t}^i = (\alpha_{1:t-1}^{I^i}, \alpha_t^i)$
  Set $\Omega_t = \sum_{i=1}^N \omega_t^i$
end for

Algorithm: Bootstrap Particle Filter

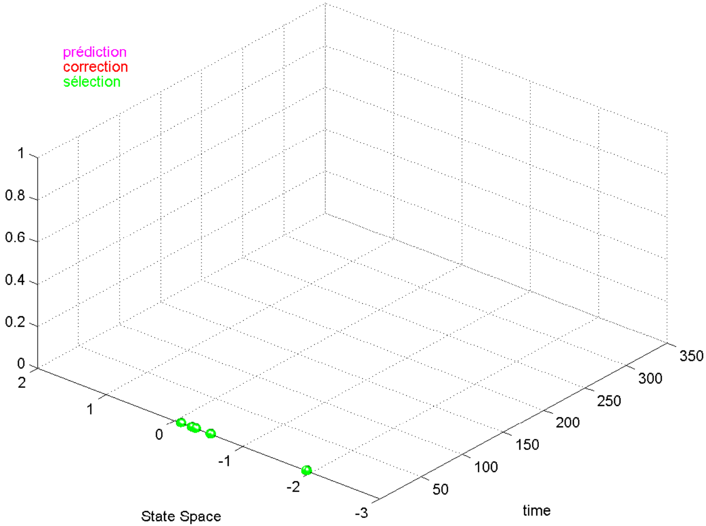
Bootstrap Particle Filter:

Draw $\alpha_1^i \sim q(\alpha_1)$
Set $\omega_1^i = g(y_1 \mid \alpha_1^i)$
Set $\Omega_1 = \sum_{i=1}^N \omega_1^i$
for $t = 2, 3, \ldots, n$ **do**
  Draw $I^i = j$ w. pr. $\frac{\omega_{t-1}^j}{\Omega_{t-1}}$
  Draw $\alpha_t^i \sim q(\alpha_t \mid \alpha_{t-1}^{I^i})$
  Set $\omega_t^i = g(y_t \mid \alpha_t^i)$
  Set $\alpha_{1:t}^i = (\alpha_{1:t-1}^{I^i}, \alpha_t^i)$
  Set $\Omega_t = \sum_{i=1}^N \omega_t^i$
end for

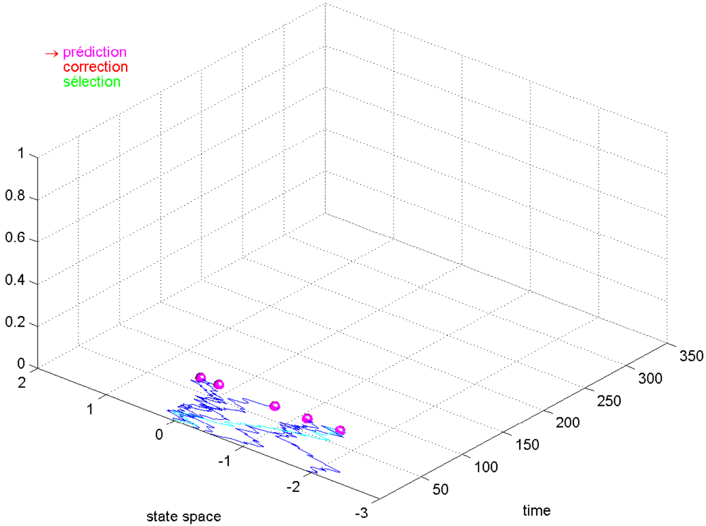
In Python:

```
part = np.zeros((n,N))
logwgt = np.zeros((n,N))
wgt = np.zeros((n,N))
ests = np.zeros(n)
part[0,:] = np.random.randn(N)
logwgt[0,:] = logwgtfun(xpart[0,:],y[0])
wgt[0,:] = np.exp(logwgt[0,:])
ests[0] = np.sum(wgt[0,:] * part[0,:])/np.sum(wgt[i+1,:])
for i in range(n-1):
    ind = np.random.choice(N, size=N, replace=True, p=wgt[i,:])
    part[i+1,:] = a*part[i,ind] + np.random.randn(N)
    logwgt[i+1,:] = logwgtfun(xpart[i+1,:],y[i+1])
    wgt[i+1,:] = np.exp(logwgt[i+1,:])
    ests[i+1] = np.sum(wgt[i+1,:] * part[i+1,:])/np.sum(wgt[i+1,:])
```

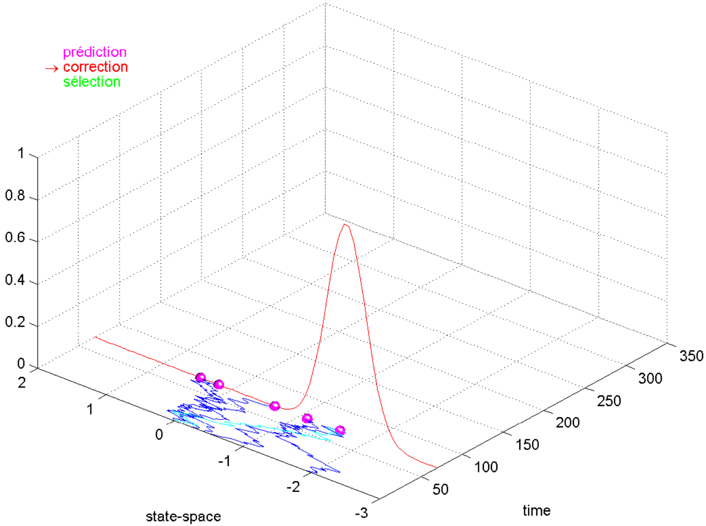
Particle Filter Movie



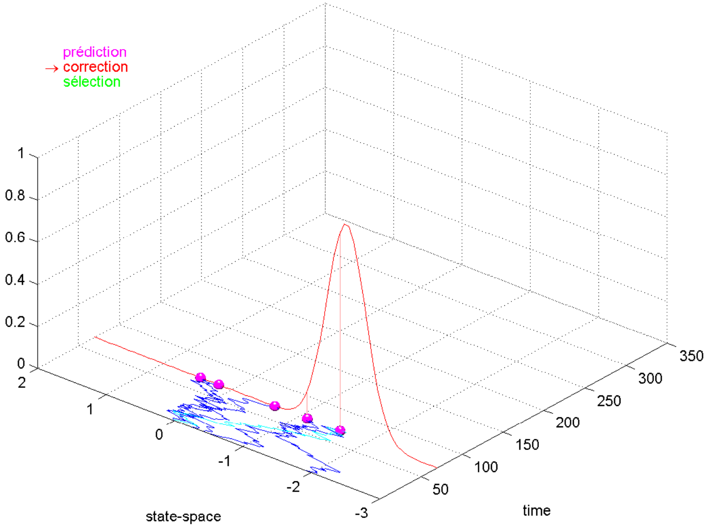
Particle Filter Movie



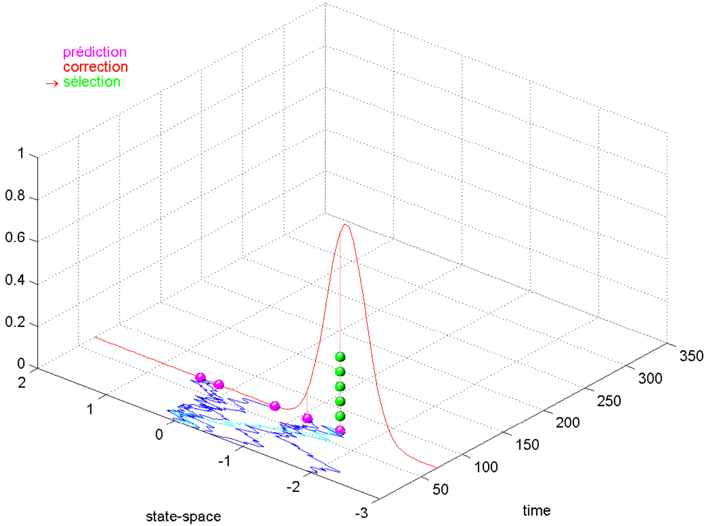
Particle Filter Movie



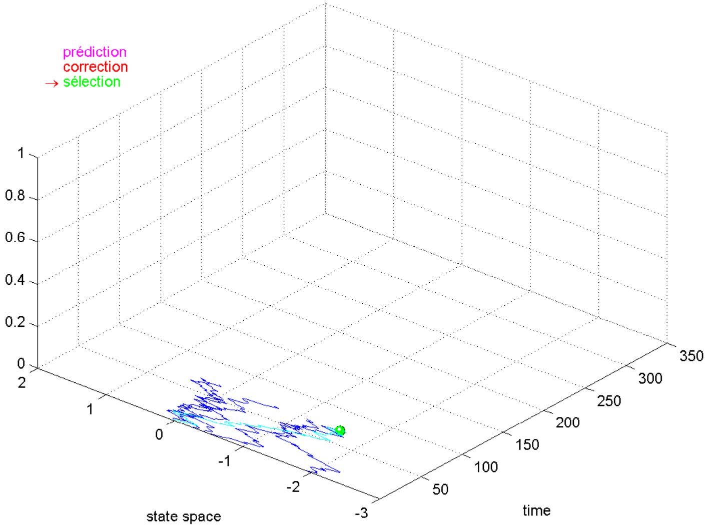
Particle Filter Movie



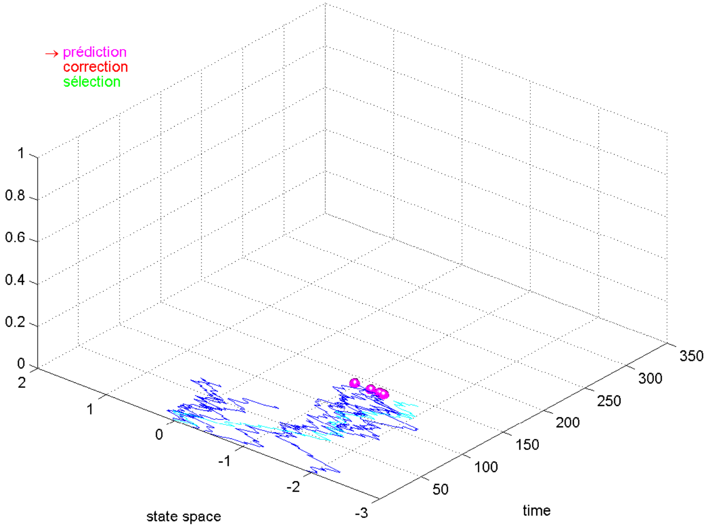
Particle Filter Movie



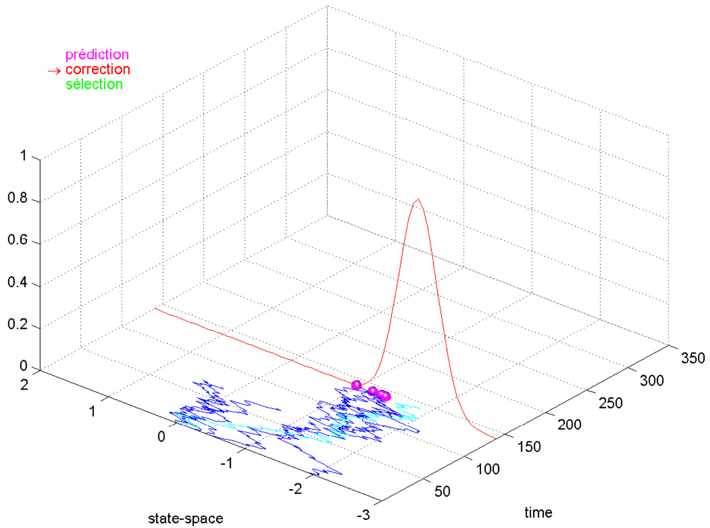
Particle Filter Movie



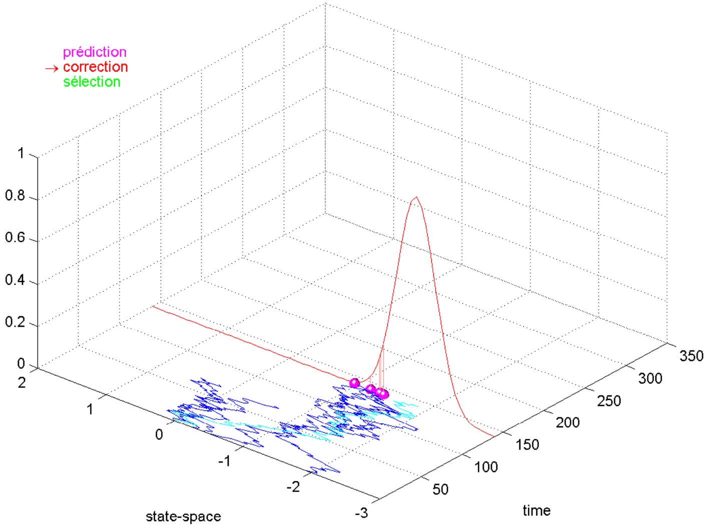
Particle Filter Movie



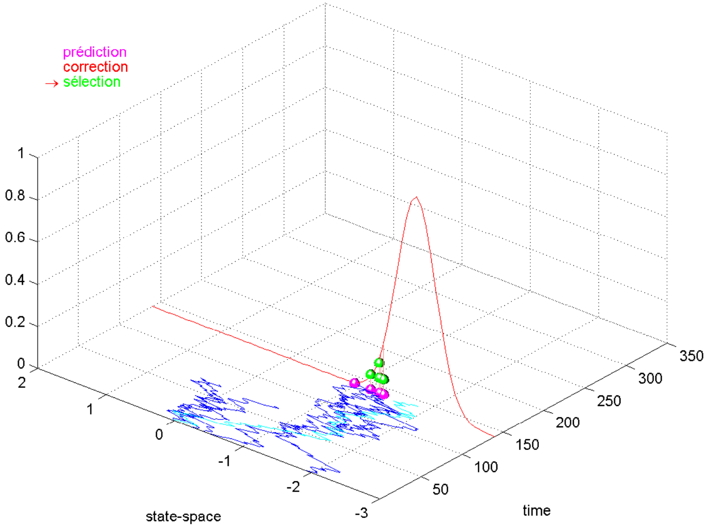
Particle Filter Movie



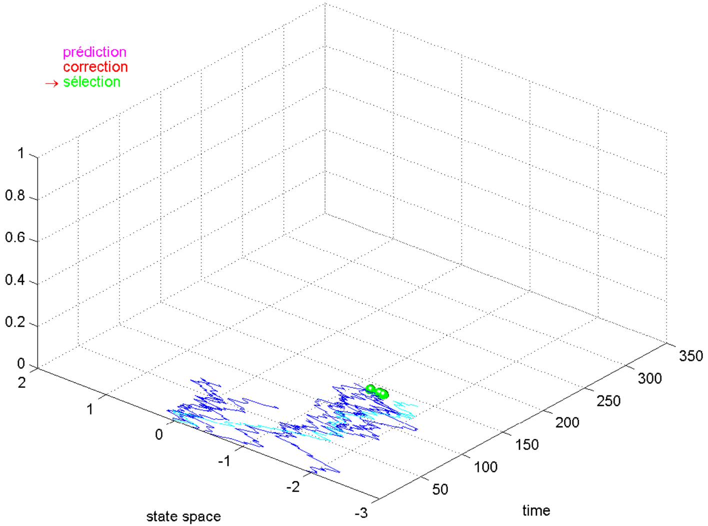
Particle Filter Movie



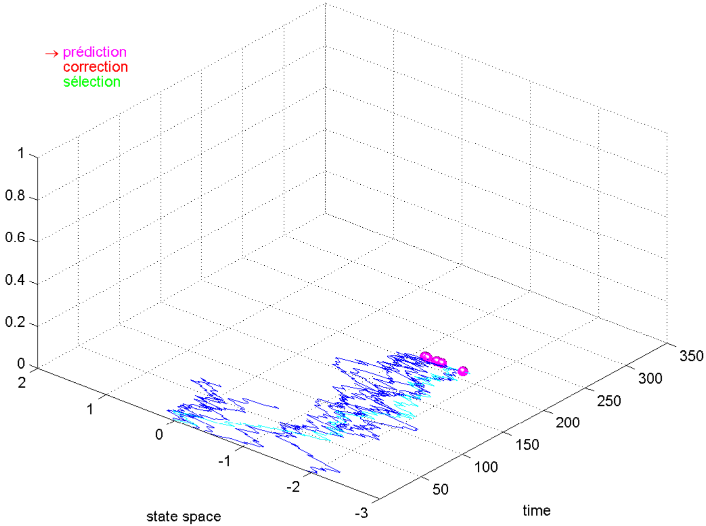
Particle Filter Movie



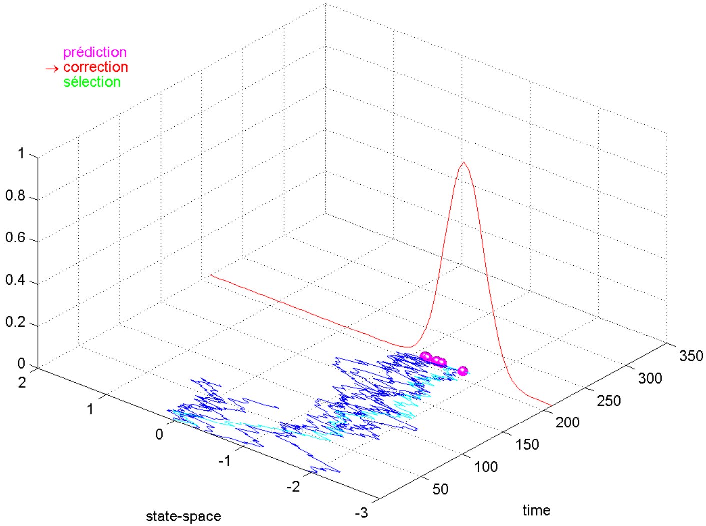
Particle Filter Movie



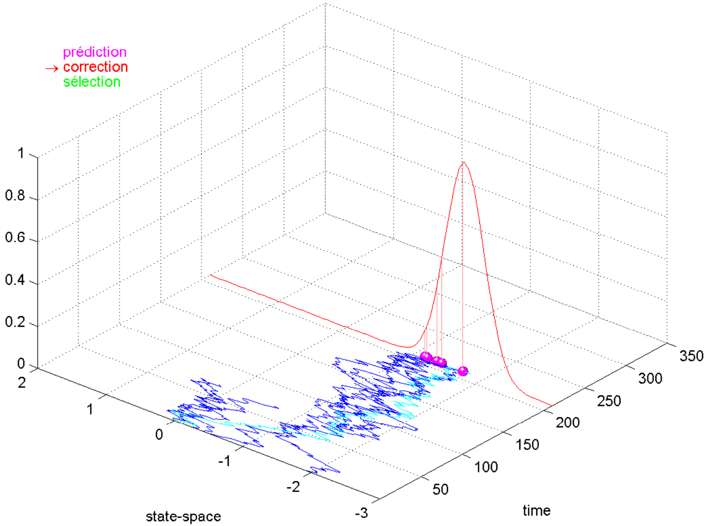
Particle Filter Movie



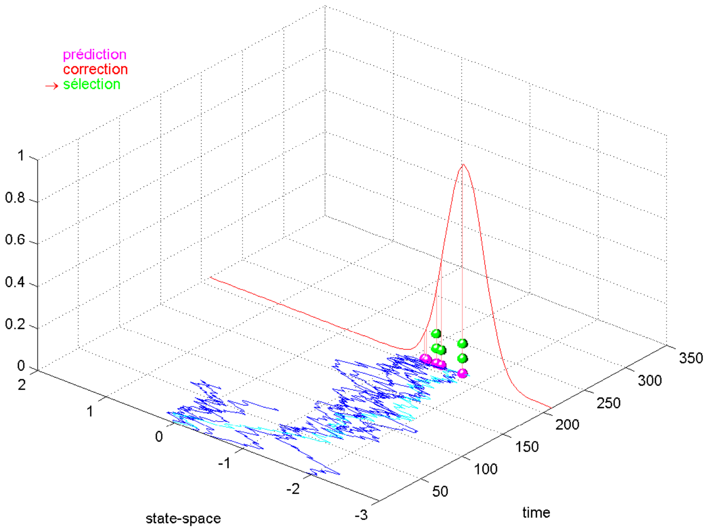
Particle Filter Movie



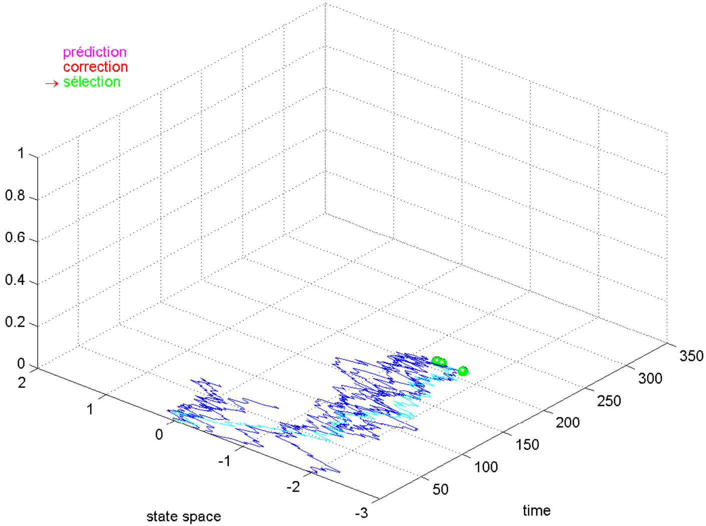
Particle Filter Movie



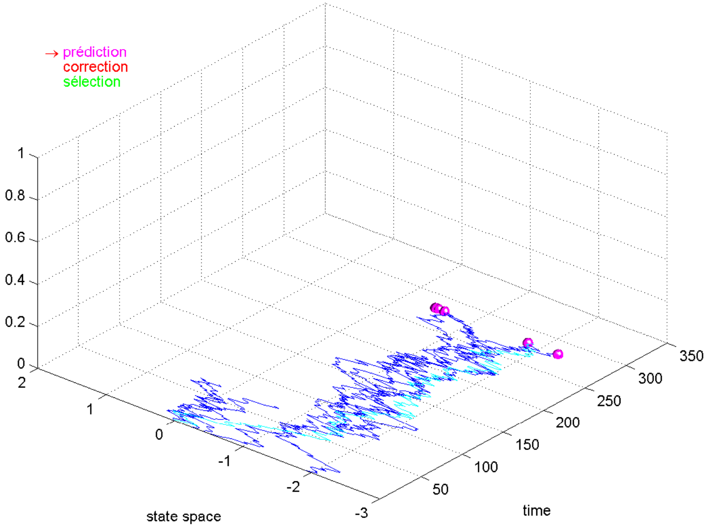
Particle Filter Movie



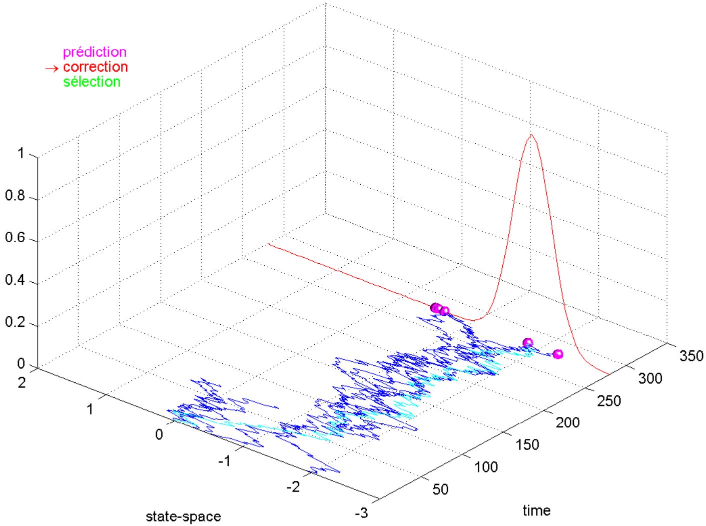
Particle Filter Movie



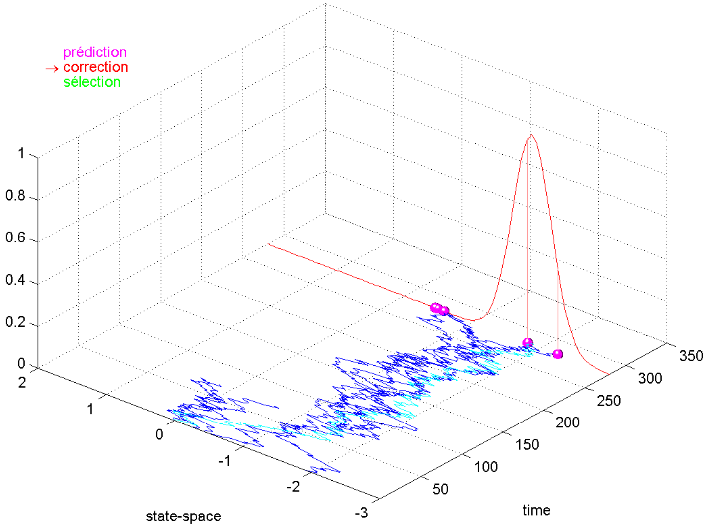
Particle Filter Movie



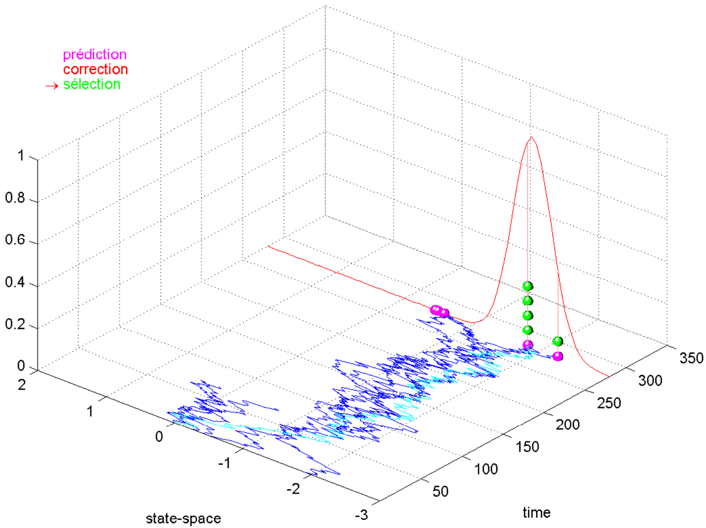
Particle Filter Movie



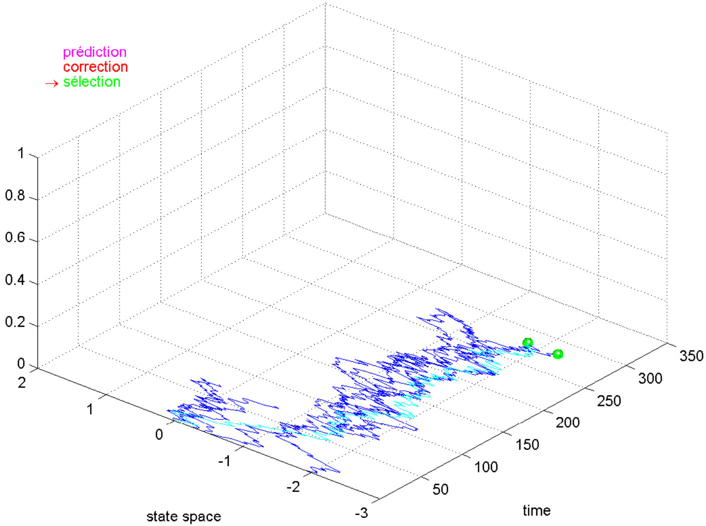
Particle Filter Movie



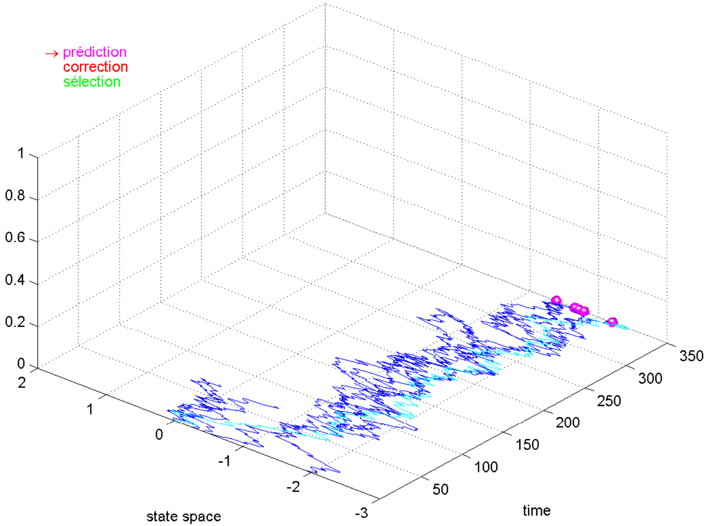
Particle Filter Movie



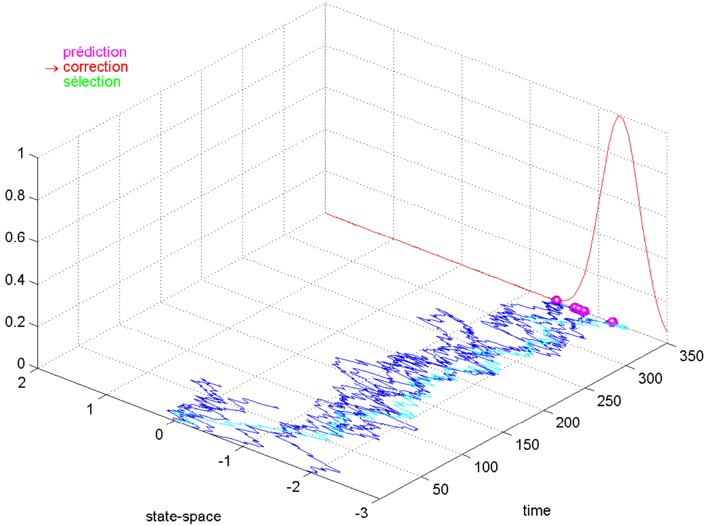
Particle Filter Movie



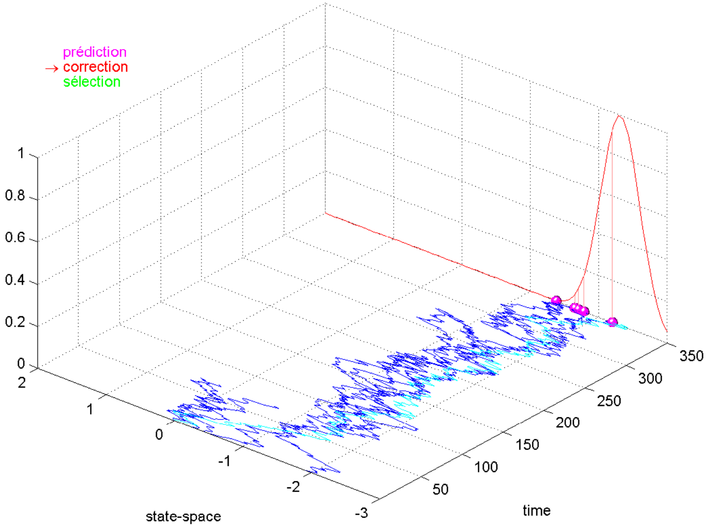
Particle Filter Movie



Particle Filter Movie



Particle Filter Movie



Parameter Estimation in General State-Space Models

Calculating the Log-Likelihood

- For the Sequential Importance Sampling algorithm we have that

$$\frac{1}{N}\sum_{i=1}^{N}\omega_n^i \approx L(y_{1:n})$$

- When adding resampling this is not a valid estimator for the likelihood.

Calculating the Log-Likelihood

- For the Sequential Importance Sampling algorithm we have that

$$\frac{1}{N}\sum_{i=1}^{N}\omega_n^i \approx L(y_{1:n})$$

- When adding resampling this is not a valid estimator for the likelihood.
- Instead we use

$$\prod_{t=1}^{n}\left(\frac{1}{N}\sum_{i=1}^{N}\omega_t^i\right) \approx L(y_{1:n})$$

- Note: ω_t^i are the unnormalized weights!

Calculating the Log-Likelihood

- For the Sequential Importance Sampling algorithm we have that

$$\frac{1}{N}\sum_{i=1}^{N}\omega_n^i \approx L(y_{1:n})$$

- When adding resampling this is not a valid estimator for the likelihood.
- Instead we use

$$\prod_{t=1}^{n}\left(\frac{1}{N}\sum_{i=1}^{N}\omega_t^i\right) \approx L(y_{1:n})$$

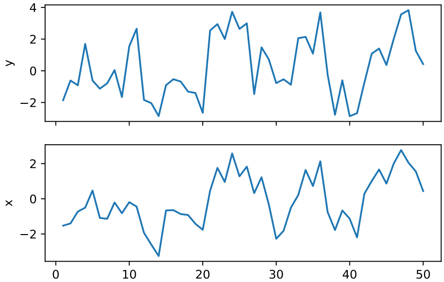
- Note: ω_t^i are the unnormalized weights!
- The is an unbiased estimator of the likelihood!

Example: Gaussian State Space Model

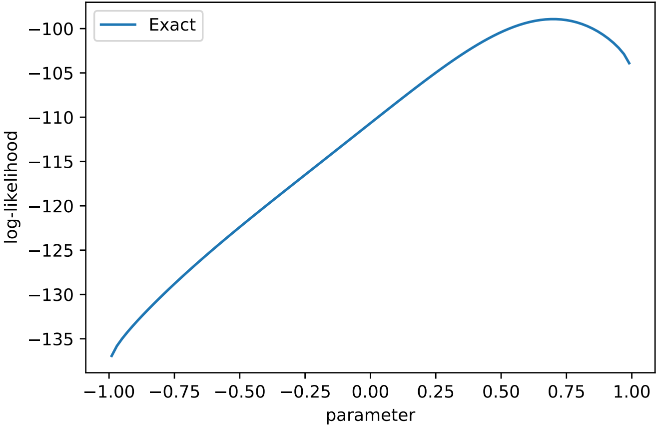
Linear Gaussian State Space Model We look at the model defined by the following equations:

$$\begin{cases} \alpha_{t+1} = a\alpha_t + \eta_{t+1}, & \eta_{t+1} \sim \mathcal{N}(0,1), \\ y_t = \alpha_t + \varepsilon_t, & \varepsilon_t \sim \mathcal{N}(0,1), \end{cases}$$

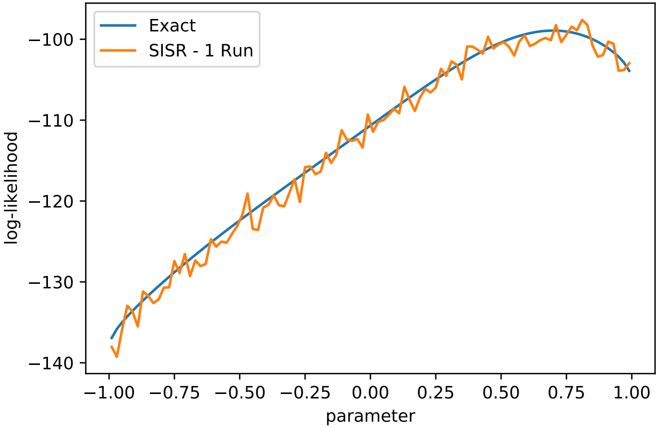
with initial distribution $\alpha_1 \sim \mathcal{N}(0, 1/(1-a^2))$



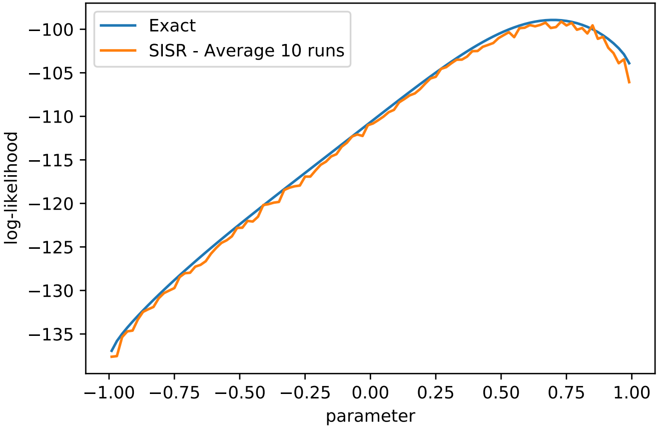
Example: Gaussian State Space Model



Example: Gaussian State Space Model



Example: Gaussian State Space Model



Expectation-Maximization

Expectation-Maximization Algorithm

- As in the LGSS model we can use expectation-maximization algorithm for parameter estimation.

Expectation-Maximization Algorithm

- As in the LGSS model we can use expectation-maximization algorithm for parameter estimation.
- The E-step is calculating:

$$\begin{aligned}
\mathcal{Q}(\tilde{\theta}, \theta) &= \mathbb{E}[\log p_\theta(\alpha_{1:n}, y_{1:n}) \mid y_{1:n}, \tilde{\theta}] \\
&= \mathbb{E}[\log q_\theta(\alpha_1) \mid y_{1:n}, \tilde{\theta}] + \sum_{t=2}^{n} \mathbb{E}[\log q_\theta(\alpha_t \mid \alpha_{t-1}) \mid y_{1:n}, \tilde{\theta}] \\
&+ \sum_{t=1}^{n} \mathbb{E}[\log g_\theta(y_t \mid \alpha_t) \mid y_{1:n}, \tilde{\theta}]
\end{aligned}$$

Expectation-Maximization Algorithm

- As in the LGSS model we can use expectation-maximization algorithm for parameter estimation.
- The E-step is calculating:

$$
\begin{aligned}
\mathcal{Q}(\tilde{\theta}, \theta) &= \mathbb{E}[\log p_\theta(\alpha_{1:n}, y_{1:n}) \mid y_{1:n}, \tilde{\theta}] \\
&= \mathbb{E}[\log q_\theta(\alpha_1) \mid y_{1:n}, \tilde{\theta}] + \sum_{t=2}^{n} \mathbb{E}[\log q_\theta(\alpha_t \mid \alpha_{t-1}) \mid y_{1:n}, \tilde{\theta}] \\
&+ \sum_{t=1}^{n} \mathbb{E}[\log g_\theta(y_t \mid \alpha_t) \mid y_{1:n}, \tilde{\theta}]
\end{aligned}
$$

- The M-step is to maximize $\mathcal{Q}(\tilde{\theta}, \theta)$:

$$\theta^* = \arg\max_\theta \mathcal{Q}(\tilde{\theta}, \theta)$$

Def: A distribution belongs to the exponential family if its density function can be written as,

$$p_\theta(x) = h(x) \exp\left(\mathsf{n}(\theta) \cdot \mathsf{T}(x) - A(\theta)\right),$$

where

- $\mathsf{T}(x)$ is a sufficient statistic
- $\mathsf{n}(\theta)$ is the natural parameter

Interlude I: Exponential Family

Def: A distribution belongs to the exponential family if its density function can be written as,

$$p_\theta(x) = h(x) \exp\left(\mathsf{n}(\theta) \cdot \mathsf{T}(x) - A(\theta)\right),$$

where

- $\mathsf{T}(x)$ is a sufficient statistic
- $\mathsf{n}(\theta)$ is the natural parameter

Ex: A Gaussian distribution with mean μ and variance σ^2 belongs to the exponential family using

$$\underbrace{\frac{1}{\sqrt{2\pi}}}_{h(x)} \exp\left(\underbrace{\begin{pmatrix} \frac{\mu}{\sigma^2} \\ -\frac{1}{2\sigma^2} \end{pmatrix}}_{\mathsf{n}(\theta)} \cdot \underbrace{\begin{pmatrix} x \\ x^2 \end{pmatrix}}_{\mathsf{T}(x)} - \underbrace{\left(\frac{\mu^2}{2\sigma^2} + \frac{1}{2}\log(\sigma^2) \right)}_{A(\theta)} \right)$$

Expectation-Maximization Algorithm II

- We assume that q_θ and g_θ belongs to the exponential family.

- We assume that q_θ and g_θ belongs to the exponential family.
- The E-step now reduces to,

$$
\begin{aligned}
\mathcal{Q}(\tilde{\theta}, \theta) = \mathbb{E}[\log h_q^1(\alpha_1) + \mathbf{n}_\mathbf{q}^1(\theta) \cdot \mathbf{T}_\mathbf{q}(\alpha_1) - A_q^1(\theta) \mid y_{1:n}, \tilde{\theta}] \\
+ \sum_{t=2}^{n} \mathbb{E}[\log h_q(\alpha_t, \alpha_{t-1}) + \mathbf{n}_\mathbf{q}(\theta) \cdot \mathbf{T}_\mathbf{q}(\alpha_t, \alpha_{t-1}) - A_q(\theta) \mid y_{1:n}\tilde{\theta}] \\
+ \sum_{t=1}^{n} \mathbb{E}[\log h_g(y_t, \alpha_t) + \mathbf{n}_\mathbf{g}(\theta) \cdot \mathbf{T}_\mathbf{g}(y_t, \alpha_t) - A_g(\theta) \mid y_{1:n}, \tilde{\theta}]
\end{aligned}
$$

Expectation-Maximization Algorithm II

- We assume that q_θ and g_θ belongs to the exponential family.
- The E-step now reduces to,

$$
\begin{aligned}
\mathcal{Q}(\tilde{\theta}, \theta) = \mathbb{E}[\log h_q^1(\alpha_1) \mid y_{1:n}, \tilde{\theta}] + \mathbf{n}_\mathbf{q}^1(\theta) \cdot \mathbb{E}[\mathbf{T}_\mathbf{q}(\alpha_1) \mid y_{1:n}, \tilde{\theta}] - A_q^1(\theta) \\
+ \sum_{t=2}^{n} \mathbb{E}[\log h_q(\alpha_t, \alpha_{t-1}) \mid y_{1:n}, \tilde{\theta}] + \mathbf{n}_\mathbf{q}(\theta) \cdot \mathbb{E}[\mathbf{T}_\mathbf{q}(\alpha_t, \alpha_{t-1}) \mid y_{1:n}, \tilde{\theta}] \\
- A_q(\theta) \\
+ \sum_{t=1}^{n} \mathbb{E}[\log h_g(y_t, \alpha_t) \mid y_{1:n}, \tilde{\theta}] + \mathbf{n}_\mathbf{g}(\theta) \cdot \mathbb{E}[\mathbf{T}_\mathbf{g}(y_t, \alpha_t) \mid y_{1:n}, \tilde{\theta}] \\
- A_g(\theta)
\end{aligned}
$$

Expectation-Maximization Algorithm II

- We assume that q_θ and g_θ belongs to the exponential family.
- The E-step now reduces to,
  Calculate the smoothed sufficient statistics,

$$\mathsf{T}_1 = \mathbb{E}[\mathsf{T}_\mathsf{q}(\alpha_1) \mid y_{1:n}, \tilde{\theta}],$$

$$\mathsf{T}_2 = \sum_{t=2}^{n} \mathbb{E}[\mathsf{T}_\mathsf{q}(\alpha_t, \alpha_{t-1}) \mid y_{1:n}, \tilde{\theta}],$$

$$\mathsf{T}_3 = \sum_{t=1}^{n} \mathbb{E}[\mathsf{T}_\mathsf{g}(y_t, \alpha_t) \mid y_{1:n}, \tilde{\theta}].$$

Expectation-Maximization Algorithm II

- We assume that q_θ and g_θ belongs to the exponential family.
- The E-step now reduces to,
  Calculate the smoothed sufficient statistics,

$$\mathsf{T}_1 = \mathbb{E}[\mathsf{T}_\mathsf{q}(\alpha_1) \,|\, y_{1:n}, \tilde{\theta}],$$

$$\mathsf{T}_2 = \sum_{t=2}^{n} \mathbb{E}[\mathsf{T}_\mathsf{q}(\alpha_t, \alpha_{t-1}) \,|\, y_{1:n}, \tilde{\theta}],$$

$$\mathsf{T}_3 = \sum_{t=1}^{n} \mathbb{E}[\mathsf{T}_\mathsf{g}(y_t, \alpha_t) \,|\, y_{1:n}, \tilde{\theta}].$$

- The M-step becomes maximize,

$$\mathsf{n}_\mathsf{q}^1(\theta) \cdot \mathsf{T}_1 - A_q^1(\theta) + \mathsf{n}_\mathsf{q}(\theta) \cdot \mathsf{T}_2 - A_q(\theta) + \mathsf{n}_\mathsf{g}(\theta) \cdot \mathsf{T}_3 - A_g(\theta)$$

Smoothing

Joint-Smoothing Distribution

- As previously for the EM-algorithm we need the smoothing distribution.

Joint-Smoothing Distribution

- As previously for the EM-algorithm we need the smoothing distribution.
- Good news! When we derived the particle filter we did it for the joint-smoothing distribution.

Joint-Smoothing Distribution

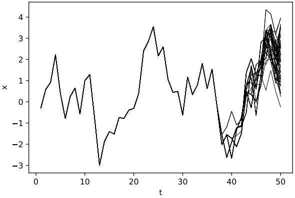
- As previously for the EM-algorithm we need the smoothing distribution.
- Good news! When we derived the particle filter we did it for the joint-smoothing distribution.
- The particle trajectories approximate the joint-smoothing distribution.

Joint-Smoothing Distribution

- As previously for the EM-algorithm we need the smoothing distribution.
- Good news! When we derived the particle filter we did it for the joint-smoothing distribution.
- The particle trajectories approximate the joint-smoothing distribution.
- Idea: Run the the particle filter, store the trajectories and approximate the smoothing distribution.

Joint-Smoothing Distribution

- As previously for the EM-algorithm we need the smoothing distribution.
- Good news! When we derived the particle filter we did it for the joint-smoothing distribution.
- The particle trajectories approximate the joint-smoothing distribution.
- Idea: Run the the particle filter, store the trajectories and approximate the smoothing distribution.
- Problem: Resampling collapses the trajectories.



Fixed-Lag Smoothing

- There are many algorithms to perform smoothing in general State Space models.
- Most of them involve intricate backward passes or Markov Chains to estimate the smoothing distribution.

Fixed-Lag Smoothing

- There are many algorithms to perform smoothing in general State Space models.
- Most of them involve intricate backward passes or Markov Chains to estimate the smoothing distribution.
- A simple smoothing algorithm is the fixed-lag smoothing.
- We approximate using

$$\mathbb{E}[h(\alpha_t) \mid y_{1:n}] \approx \mathbb{E}[h(\alpha_t) \mid y_{1:t+\ell}],$$

where ℓ is a fixed lag.

Example: Stochastic Volatility

Stochastic volatility model. The model is defined by the equations,

$$\begin{cases} \alpha_k = a\alpha_{k-1} + \sigma_\eta \eta_k, & \eta_k \sim \mathcal{N}(0, s^2) \\ y_k = b \exp(\alpha_k/2)\varepsilon_k, & \varepsilon_k \sim \mathcal{N}(0, 1) \end{cases}$$

For simplicity we assume that $\alpha_1 \sim \mathcal{N}(0, 1)$.

Example: Stochastic Volatility

EM-algorithm, in this model we find four sufficient statistics:

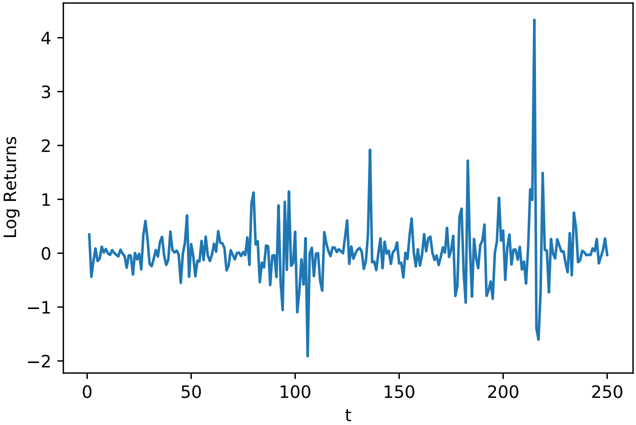
$$t_1 = \sum_{t=2}^{n} \mathbb{E}[\alpha_t^2 \mid y_{1:n}] \qquad t_2 = \sum_{t=2}^{n} \mathbb{E}[\alpha_t \alpha_{t-1} \mid y_{1:n}]$$

$$t_3 = \sum_{t=2}^{n} \mathbb{E}[\alpha_{t-1}^2 \mid y_{1:n}] \qquad t_4 = \sum_{t=1}^{n} \mathbb{E}[y_t^2 \exp(-\alpha_t) \mid y_{1:n}]$$

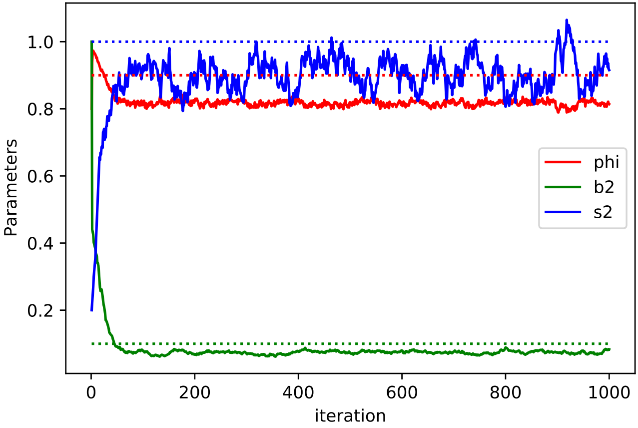
The parameters are updated as,

$$a = \frac{t_2}{t_3} \quad s^2 = \frac{1}{n-1}\left(t_1 - \frac{t_2^2}{t_3}\right) \quad b^2 = \frac{t_4}{n}$$

Example: Stochastic Volatility



Example: Stochastic Volatility



Extensions to Particle Filters

Adaptive Resampling

- So far we have done resampling every iteration.
- This is unnecessary, we only need to do it if the weights have degenerated.

Adaptive Resampling

- So far we have done resampling every iteration.
- This is unnecessary, we only need to do it if the weights have degenerated.
- Use effective sample size (ESS) as a measure for degeneracy,

$$\mathrm{ESS}_t = \frac{(\sum_{i=1}^{N} \omega_t^i)^2}{\sum_{i=1}^{N} (\omega_t^i)^2}$$

Adaptive Resampling

- So far we have done resampling every iteration.
- This is unnecessary, we only need to do it if the weights have degenerated.
- Use effective sample size (ESS) as a measure for degeneracy,

$$\mathrm{ESS}_t = \frac{(\sum_{i=1}^{N} \omega_t^i)^2}{\sum_{i=1}^{N} (\omega_t^i)^2}$$

- If all weights are equal then $\mathrm{ESS}_t = N$.

Adaptive Resampling

- So far we have done resampling every iteration.
- This is unnecessary, we only need to do it if the weights have degenerated.
- Use effective sample size (ESS) as a measure for degeneracy,

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

- If all weights are equal then $\mathrm{ESS}_t = N$.
- If all weights except one is zero then $\mathrm{ESS}_t = 1$.

Adaptive Resampling

- So far we have done resampling every iteration.
- This is unnecessary, we only need to do it if the weights have degenerated.
- Use effective sample size (ESS) as a measure for degeneracy,

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

- If all weights are equal then $\mathrm{ESS}_t = N$.
- If all weights except one is zero then $\mathrm{ESS}_t = 1$.
- Choose a threshold N_{ESS}, if $\mathrm{ESS}_t < N_{\mathrm{ESS}}$ then resample, otherwise no resampling happens.
- If no resampling happens the weights should be updated as in the SIS algorithm.

Algorithm: Particle Filter with Adaptive Resampling

Particle Filter:

Draw $\alpha_1^i \sim f(\alpha_1)$

Set $\omega_1^i = \frac{q(\alpha_1^i)g(y_1 \mid \alpha_1^i)}{f(\alpha_1^i)}$

Set $\Omega_1 = \sum_{i=1}^N \omega_1^i$

for $t = 2, 3, \ldots, n$ **do**

  Calculate $\mathrm{ESS}_{t-1} = \frac{(\sum_{i=1}^N \omega_t^i)^2}{\sum_{i=1}^N (\omega_t^i)^2}$

  if $\mathrm{ESS}_{t-1} < N_{\mathrm{ESS}}$ **then**

    Draw $I^i = j$ w. pr. $\frac{\omega_{t-1}^j}{\Omega_{t-1}}$

    Set $\alpha_{1:t}^i = (\alpha_{1:t-1}^{I^i}, \alpha_t^i)$

    Set $\omega_{t-1}^i = 1/N$

  end if

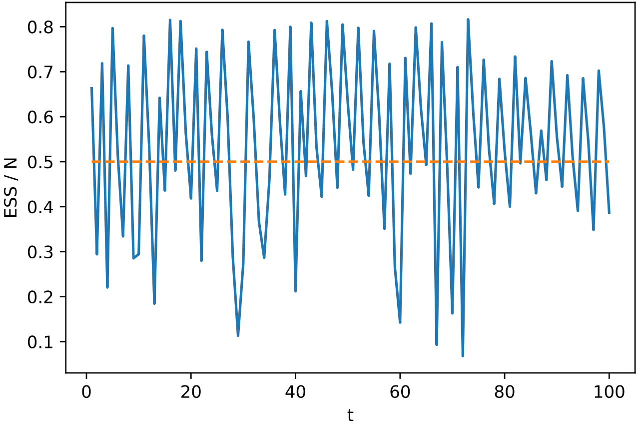
  Draw $\alpha_t^i \sim f(\alpha_t \mid \alpha_{1:t-1}^i)$

  Set $\omega_t^i = \frac{q(\alpha_t^i \mid \alpha_{t-1}^i)g(y_t \mid \alpha_t^i)}{f(\alpha_t^i \mid \alpha_{1:t-1}^i)} \times \omega_{t-1}^i$

  Set $\Omega_t = \sum_{i=1}^N \omega_t^i$

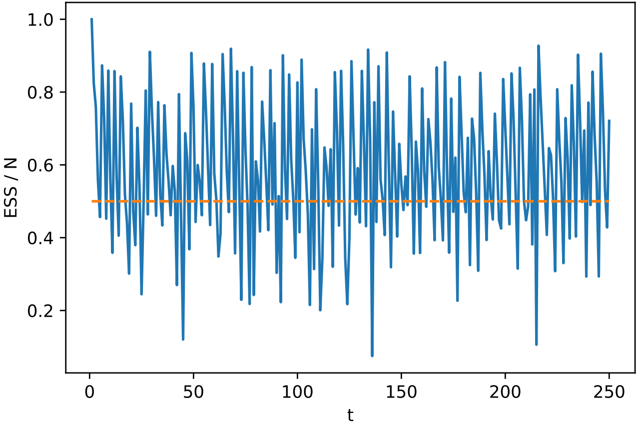
end for

Example: Adaptive resampling



Resampling: 43%

Example: Adaptive resampling



Resampling: 36%

Different Resampling Schemes

- We have so far only considered multinomial resampling.
- There are many different schemes,
    - Residual resampling

Different Resampling Schemes

- We have so far only considered multinomial resampling.
- There are many different schemes,
  - Residual resampling
  - Stratified resampling

Different Resampling Schemes

- We have so far only considered multinomial resampling.
- There are many different schemes,
  - Residual resampling
  - Stratified resampling
  - Systematic resampling

Different Resampling Schemes

- We have so far only considered multinomial resampling.
- There are many different schemes,
  - Residual resampling
  - Stratified resampling
  - Systematic resampling

Summary

- We looked at how to calculate the likelihood using a particle filter.
  - The randomness of the algorithm gives us noisy likelihood estimate.
  - Solve by performing many runs and average.

Summary

- We looked at how to calculate the likelihood using a particle filter.
    - The randomness of the algorithm gives us noisy likelihood estimate.
    - Solve by performing many runs and average.
- We looked at the EM-algorithm.
    - Easier if the model belongs to the exponential family.
    - Requires, again, the smoothing distribution.
    - Looked at the fixed-lag smoother.

Summary

- We looked at how to calculate the likelihood using a particle filter.
    - The randomness of the algorithm gives us noisy likelihood estimate.
    - Solve by performing many runs and average.
- We looked at the EM-algorithm.
    - Easier if the model belongs to the exponential family.
    - Requires, again, the smoothing distribution.
    - Looked at the fixed-lag smoother.
- We looked at some extensions regarding the resampling.
    - By calculating the ESS we performed adaptive resampling by only resampling when needed.
    - We looked at some different resampling schemes.