Markov Chain Monte Carlo Methods

Tutorial for KTH PGM DD2424

Report format

The report for this tutorial should be a single PDF file. Pen-and-paper exercises can be done in LaTeX or as scanned hand-written notes. For all coding exercises, include plots, answers to the questions, and your code.

1 Introduction and Overview

Markov Chain Monte Carlo techniques have been developed to address high-dimensional integration and optimisation problems. [1] outlines three main problem classes in machine learning and Bayesian inference that can be addressed by MCMC:

1. Enable sampling from posterior $p(x|y)$, assuming access to likelihood $p(y|x)$ and prior $p(x)$:
$$p(x|y) = \frac{p(y|x)p(x)}{\int_{x' \in \mathcal{X}} p(y|x')p(x')dx'}$$
2. Compute marginals $p(x|y) = \int_{z \in \mathcal{Z}} p(x, z|y)dz$ given the joint posterior of $(x, z) \in \mathcal{X} \times \mathcal{Z}$
3. Compute expectations of some function of interest $f : E_{p(x|y)}\big[f(x)\big] = \int_{\mathcal{X}} f(x)p(x|y)dx$

Further applications of MCMC include various problems in statistical mechanics, optimization, probabilistic model selection and simulation of physical systems.

1.1 Monte Carlo Principle

Let's focus on the last of the three problems – computing expectations of some function of interest f under the distribution $p(x|y)$. One way to approximate integral $I(f) = \int_{\mathcal{X}} f(x)p(x|y)dx$ could be to define a grid of points from the domain $\mathcal{X}$ and compute an estimate of the integral using $\sum_{x^{(i)} \in Grid} f(x^{(i)})p(x^{(i)}|y)$.

Unfortunately, unless we grow the number of our grid points exponentially, in high dimensions most of our grid samples would cover only a very small part of the space. Hence, they would likely miss the regions where the density $p(x^{(i)}|y)$ is high.

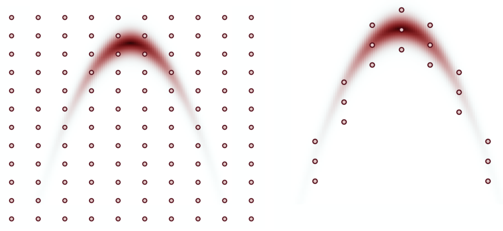


Figure 1: Monte Carlo principle can help alleviate the curse of dimensionality: in high dimensions computations using a regular grid could be wasteful (top); using samples from posterior can help focus the computation on the typical set – the region of high density of $p(x|y)$.

The idea of Monte Carlo simulation is to draw samples from the target distribution $p(x|y)$ and then approximate the expectation of f by a sum: $I_N(f) = \frac{1}{N}\sum_{i=1}^{N} f(x^{(i)}) \underset{N\to\infty}{\approx} \int_{\mathcal{X}} f(x)p(x|y)dx = I(f)$. Such estimate $I_N(f)$ is unbiased, and by law of large numbers with converge almost surely to $I(f)$ as $N \to \infty$. The error $I_N(f) - I(f)$ will converge in distribution to $\mathcal{N}\Big(0, \frac{Var(f)}{\sqrt{N}}\Big)$ by the central limit theorem.

1.2 Markov Chain Monte Carlo Principle

Ordinary Monte Carlo principle described above can not be applied to settings when we can not draw samples from the posterior $p(x|y)$ directly. Markov Chain Monte Carlo (MCMC) is a strategy for drawing samples $x^{(i)}$ that mimic samples from $p(x|y)$ while only requiring ability to evaluate $p(x|y)$ up to a normalizing constant (e.g. $p(x|y) \propto p(y|x)p(x)$).

A stochastic process composed of a family of random variables $x^{(i)}$ is called a Markov chain if $p(x^{(i)}|x^{(i-1)}, ..., x_1) = T(x^{(i)}|x^{(i-1)})$. More generally, the term 'Markov' frequently refers to the memoryless property of a certain process or model. In the case of MCMC the Markov property implies that each $x^{(i)}$ can be generated using a stochastic matrix T and the previous term in the chain $x^{(i-1)}$. The chain is called homogeneous if the same transition matrix T is used for all i.

For concreteness, let's consider a Markov chain with 3 states, a transition matrix T and initial state distribution p_0 as follows:

$$T = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0.1 & 0.9 \\ 0.6 & 0.4 & 0 \end{bmatrix} \quad p_0(x^{(1)}) = (0.5, 0.2, 0.3)$$

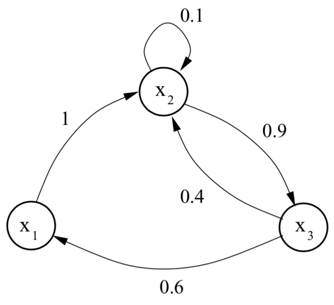


Figure 2: Transition graph for the Markov chain with $\mathcal{X} = \{x_1, x_2, x_3\}$ from [1]

After the first transition (multiplication by T) we get $p_0(x^{(1)})T = (0.2, 0.6, 0.2)$. After further transitions $p_0(x^{(1)})T^t$ converges to $p(x) = (0.2, 0.4, 0.4)$. It can be shown that applying T iteratively to any initial distribution will cause the chain to converge to $p(x) = (0.2, 0.4, 0.4)$.

To ensure that $p(x)$ is the invariant distribution for a stochastic matrix T is is sufficient to show that:

$$p(x^{(i)})T(x^{(i-1)}|x^{(i)}) = p(x^{(i-1)})T(x^{(i)}|x^{(i-1)}) \tag{1}$$

There is still a possibility that this invariant distribution is not unique. A stochastic transition matrix T will have a unique invariant distribution $p(x)$ if T satisfies we following two properties:

1. Irreducibility: there is a positive probability of visiting all other states, regardless of the starting state of the Markov chain. This is the same as stating that the transition graph is connected.

2. Aperiodicity. The chain does not get trapped in cycles. Meaning, we do not get exactly the same probability distribution vector after obtaining it earlier in the chain.

These two properties imply an 'ergodic' Markov chain and insure the uniqueness of any invariant distribution. Equation 1 is called the 'detailed balance' condition. A Markov chain that has such a distribution is called reversible. Equation 1 is used to construct one of the most widely-used MCMC approaches – the Metropolis-Hastings algorithm. Note that this condition is not necessary, meaning there are other ways to construct (advanced) MCMC algorithms without a T that explicitly satisfies the detailed balance condition.

Using insights from linear algebra and spectral theory, we can observe that $p(x)$ is the left eigenvector of T with unit eigenvalue. The remaining eigenvalues have absolute value < 1; the second largest eigenvalue determines the rate of convergence of the chain. In contrast, T lacking aperiodicity would have more than one eigenvector with eigenvalue 1. In continuous state spaces, T becomes an integral kernel K and $p(x)$ becomes the corresponding eigenfunction: $p(x^{(i)}) = \int p(x^{(i-1)})K(x^{(i)}|x^{(i-1)})dx^{(i-1)}$. The kernel K is the conditional density of $x^{(i)}$ given the value $x^{(i-1)}$.

2 Classical Monte Carlo sampling methods

2.1 Rejection sampling

Rejection sampling is one of the basic Monte Carlo sampling methods. It is used to draw samples from a distribution $p(x)$, which is known up to a normalizing constant, by sampling from another easy-to-sample distribution distribution $q(x)$. It is important, that we can evaluate the probability density function $p(x)$ and that the proposal distribution $q(x)$ satisfies:

$$p(x) \leq Mq(x), M < \infty. \tag{2}$$

The sampling procedure :

1. Set $i = 1$

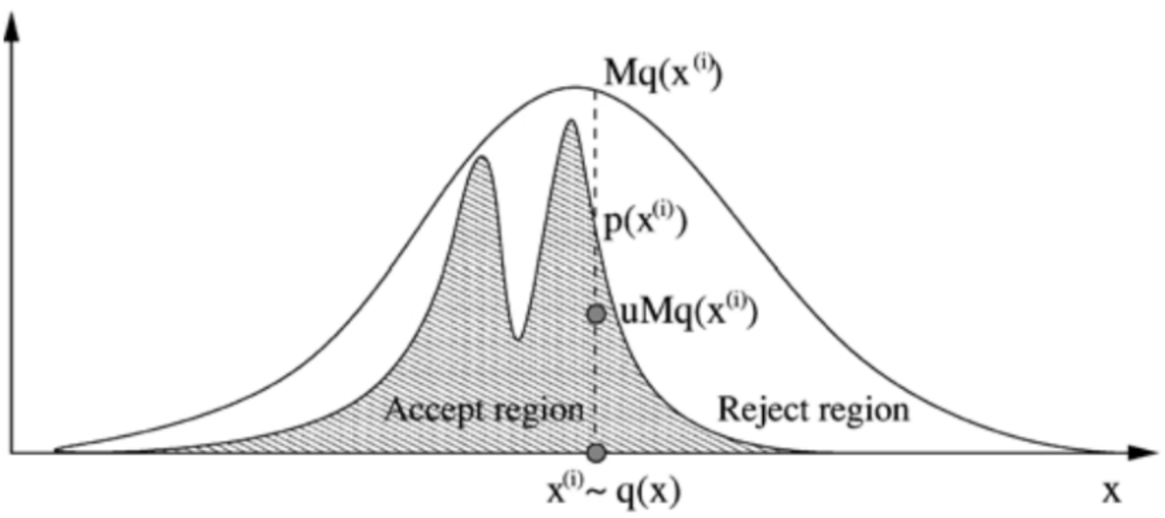


Figure 3: Rejection sampling. Figure is taken from [1].

2. Repeat until $i = N$
   (a) Sample $x^{(*)} \sim q(x)$ and $u \sim U_{(0,1)}$
   (b) If $u < \frac{p(x^{(*)})}{Mq(x^{(*)})}$, then accept $x^{(i)} = x^{(*)}$ and $i = i + 1$, else reject and continue to 2.a.
3. Return samples $\{x^{(i)}\}_{i=1}^{N}$

An illustration one step in the Rejection Sampling algorithm is shown in Figure 3.

The acceptance probability is $p(u < \frac{p(x^{(*)})}{Mq(x^{(*)})}) = \frac{1}{M}$ (Exercise).

A major limitation of the method is the need to bound $p(x)$ with $Mq(x)$. We can always find an appropriate M, but increasing M will decrease the probability of accepting samples, making the method impractical, especially for high-dimensional distributions.

2.2 Importance Sampling

Another classical method is Importance Sampling. It provides a framework for computing an expectation of a function of interest under the distribution $p(x)$: $E_{p(x)}[f(x)]$, but doesn't give a mechanism for drawing samples from $p(x)$. The standard way to deal with an untractable expectation is to use Monte Carlo (MC) estimation:

$$E_{p(x)}[f(x)] \approx \frac{1}{L}\sum_{l=1}^{L} f(x^{(l)}), \tag{3}$$

where $x^{(l)} \sim p(x)$. However, when it is impractical or impossible to sample from $p(x)$, we have to use other techniques, such as Importance Sampling.

The main idea of the method is to sample from an easy-to-sample proposal distribution $q(x)$ (that includes the support of $p(x)$, and use all the samples for MC estimation, but weight those samples according to $w(x) := \frac{p(x)}{q(x)}$.

$$E_{p(x)}[f(x)] \approx \frac{1}{L}\sum_{l=1}^{L} f(x^{(l)})w(x^{(l)}), \tag{4}$$

where $x^{(l)} \sim q(x)$.

Importance sampling can also be used when $p(x)$ can only be evaluated up to a normalizing constant $p(x) = \frac{1}{Z}\tilde{p}(x)$. In this case the estimator becomes:

$$E_{p(x)}[f(x)] \approx \sum_{l=1}^{L} f(x^{(l)})w^*(x^{(l)}), \tag{5}$$

where $w^*(x^{(l)}) = \frac{\tilde{w}(x^{(l)})}{\sum_m \tilde{w}(x^{(m)})} = \frac{\tilde{p}(x^{(l)})/q(x^{(l)})}{\sum_m \tilde{p}(x^{(m)})/q(x^{(m)})}$

2.3 Exercise (pen-and-paper) [2pts]

1. **Rejection sampling**
   (a) Show/explain why it is sufficient to know $p(x)$ up to a normalizing constant.
   (b) Rejecting too many samples will slow the sampling, Derive the acceptance probability $p(u < \frac{\tilde{p}(x)}{Mq(x)})$ for the unnormalized case, where $p(x) = \frac{\tilde{p}(x)}{Z}$.
2. **Importance sampling**
   (a) Show that $E_{p(x)}[f(x)] \approx \frac{1}{L}\sum_{l=1}^{L} f(x^{(l)})w(x^{(l)})$, where $x^{(l)} \sim q(x)$ (equation 4).
   (b) Why do we need importance weights?

3 Metropolis-Hastings Sampling

3.1 Theory

The Metropolis-Hastings (MH) algorithm is the most popular MCMC method.

At step i of the algorithm, in which the current state is $x^{(i)}$, we draw a sample x^* from the proposal conditional distribution $q(x^*|x^{(i)})$. The Markov Chain will then move to the new state x^* with probability

$$A(x^{(i)}, x^*) = min\Big\{1, \frac{p(x^*)q(x^{(i)}|x^*)}{p(x^{(i)})q(x^*|x^{(i)})}\Big\}, \tag{6}$$

otherwise it will remain in $x^{(i)}$.

The sampling procedure:

1. Initialize $x^{(0)}$
2. For $i = 0$ to $i = N - 1$
   - Sample $u \sim U_{(0,1)}$
   - Sample $x^* \sim q(x^*|x^{(i)})$
   - If $u < A(x^{(i)}, x^*) = min\Big\{1, \frac{p(x^*)q(x^{(i)}|x^*)}{p(x^{(i)})q(x^*|x^{(i)})}\Big\}$, then $x^{(i+1)} = x^*$, else $x^{(i+1)} = x^{(i)}$
3. Return samples $\{x^{(i)}\}_{i=1}^{N}$

The transition kernel for MH is

$$T(x^{(i+1)}|x^{(i)}) = q(x^{(i+1)}|x^{(i)})A(x^{(i)}, x^{(i+1)}) + \delta_{x^{(i)}}(x^{(i+1)})r(x^{(i)}), \tag{7}$$

where $r(x^{(i)}) = \int q(x^*|x^{(i)})(1 - A(x^{(i)}, x^*))dx^*$. The transition kernel satisfies the detailed balance condition (equation 1) (exercise), and, as a consequence, MH admits $p(x)$ as invariant distribution.

Notice, that the evaluation of the acceptance probability does not require knowing the normalizing constant $p(x) = \frac{1}{Z}\tilde{p}(x)$.

Special cases of MH include *Independent Sampler* and *Metropolis Algorithm.* In Independent Sampler the proposal distribution is independent of the current state, i.e. $q(x^*|x^{(i)}) = q(x^*)$. The Metropolis Algorithm assumes symmetric proposal distribution $q(x^*|x^{(i)}) = q(x^{(i)}|x^*)$, in which case the acceptance probability becomes:

$$A(x^{(i)}, x^*) = min\Big\{1, \frac{p(x^*)}{p(x^{(i)})}\Big\}, \tag{8}$$

3.2 Exercises

3.2.1 Pen-and-paper [2pts]

1. Show, that the Markov Chain transition kernel (equation 7) in Metropolis-Hastings Algorithm satisfy the detailed balance condition (equation 1).

2. Why would you want to use an asymmetrical proposal distribution, i.e MH and not just Metropolis?

3. For a model with PGM $p(y, x) = p(x)p(y|x)$, what could be a reasonable proposal distribution for the *Independent Sampler* when we want to sample from posterior $p(x|y)$?

3.2.2 Coding [4pts]

Problem formulation: We want to sample from the target distribution with unknown normalizing constant.

1. Implement Metropolis-Hastings algorithm from scratch (in your preferred language). It should work for the general case, when the normalizing constant of the target distribution is unknown.

2. Use your MH implementation to sample from a Gaussian Mixture Model
$$p(x) \propto a_1\mathcal{N}(x|\mu_1, \sigma_1^2) + a_2\mathcal{N}(x|\mu_2, \sigma_2^2) \tag{9}$$
with the following parameters: $a_1 = a_2 = 0.5, \mu_1 = 0, \sigma_1 = 1, \mu_2 = 3, \sigma_2 = 0.5$. Use a Gaussian as proposal distribution. Plot the random walk. Try different variances for the proposal distribution. Report your results (with plots). How does the proposal variance affect sampling?

4 Particle Filtering / Sequential Monte Carlo (SMC)

4.1 Motivation and Problem Statement

In settings where observations arrive sequentially, incremental updates can be done to the Bayesian posterior and inference can be performed online. If the transitions are modeled by a linear-Gaussian state space model, then exact analytical updates can be derived. The result is known as Kalman filter. If the data is discrete, an analytical solution known as HMM filter can be derived. However, Gaussian and linearity assumptions are limiting, so more flexible algorithms are needed. One such option is the family of Sequential Monte Carlo (SMC) methods. These can appear under the names of bootstrap filters, particle filters, Monte Carlo filters. These methods are highly parallelizable and widely applicable.

Suppose we have a sequence of observations $y_{1:t} = \{y_1, ..., y_t\}, y_i \in \mathcal{Y}, t \in \mathbb{N}$ that are conditionally independent given some unobserved (hidden) states $x_i \in \mathcal{X}$. Suppose that this unobserved signal $x_{0:t} = \{x_0, x_1, ..., x_t\}$ is modeled as a Markov process, with initial distribution $p(x_0)$ and transition $p(x_t|x_{t-1})$. We will aim to estimate the posterior $p(x_{0:t}|y_{1:t})$, a marginal $p(x_t|y_{1:t})$ known as filtering distribution and expectations of some function of interest $f_t : \mathcal{X} \to \mathbb{R}^n$. For example, if f_t is conditional mean, then $f_t(x_{0:t}) = x_{0:t}$; if f_t is conditional variance then $f_t(x_{0:t}) = xx_t^T - \mathbb{E}_{p(x_t|y_{1:t})}[x_T]\mathbb{E}^T_{p(x_t|y_{1:t})}[x_t]$. To summarize, we have the following setup:

$$\begin{tikzcd}
X_0 \arrow[r] \arrow[d] & X_1 \arrow[r] \arrow[d] & X_2 \arrow[r] \arrow[d] & X_3 \arrow[r] \arrow[d] & \cdots & \text{signal} \\
Y_0 & Y_1 & Y_2 & Y_3 & \cdots & \text{observation}
\end{tikzcd}$$

Figure 4: Visualization of the filtering problem (from [2])

Given :		*Need to Estimate :*	
$p(x_0)$	initial distribution	$p(x_{0:t}\|y_{1:t})$	posterior distribution
$p(x_t\|x_{t-1}), t \geq 1$	transition equation	$p(x_t\|y_{1:t})$	filtering distribution
$p(y_t\|x_t), t \geq 1$	marginal distribution	$I(f_t) = \mathbb{E}_{p(x_{0:t}\|y_{1:t})}[f_t(x_{0:t})] = \int f_t(x_{0:t})p(x_{0:t}\|y_{1:t})dx_{0:t}$	

For any time t, the posterior is given by the Bayes' theorem:

$$p(x_{0:t}|y_{1:t}) = \frac{p(y_{1:t}|x_{0:t})p(x_{0:t})}{\int p(y_{1:t}|x_{0:t})p(x_{0:t})dx_{0:t}}$$

The recursive definition of the joint distribution is:

$$p(x_{0:t+1}|y_{1:t+1}) = p(x_{0:t}|y_{1:t})\frac{p(y_{t+1}|x_{t+1})p(x_{t+1}|x_t)}{p(y_{t+1}|y_{1:t})}$$

The marginal distribution can be computed through the following recursive formula:

$$\textit{Prediction: } p(x_t|y_{1:t-1}) = \int p(x_t|x_{t-1})p(x_{t-1}|y_{1:t-1})dx_{t-1}$$

$$\textit{Updating: } p(x_t|y_{1:t}) = \frac{p(y_t|x_t)p(x_t|y_{1:t-1})}{\int p(y_t|x_t)p(x_t|y_{1:t-1})dx_t}$$

The above expressions are computationally intractable, because of the intractable normalizing constant $p(y_{1:t})$, marginals of the posterior $p(x_{0:t}|y_{1:t})$ as well as $I(f_t)$, since all these require evaluating intractable high-dimensional integrals.

4.2 Sequential Importance Sampling

We can choose an arbitrary importance sampling distribution, also called proposal distribution, $\pi(x_{0:t}|y_{1:t})$ such that its support includes the support of $p(x_{0:t}|y_{1:t})$. We can then simulate N i.i.d. particles (i.e. samples) $\{x_{0:t}^{(i)}, i = 1, ..., N\}$ from $\pi(x_{0:t}|y_{1:t})$ and obtain a Monte Carlo estimate of $I(f_t)$:

$$w(x_{0:t}) = \frac{p(x_{0:t}|y_{1:t})}{\pi(x_{0:t}|y_{1:t})} \text{ : importance weights}$$

$$I(f_t) = \frac{\int f_t(x_{0:t})w(x_{0:t})\pi(x_{0:t}|y_{1:t})dx_{0:t}}{\int w(x_{0:t})\pi(x_{0:t}|y_{1:t})dx_{0:t}} = \frac{\int f_t(x_{0:t})p(x_{0:t}|y_{1:t})dx_{0:t}}{1}$$

$$\hat{I}_N(f_t) = \frac{\frac{1}{N}\sum_{i=1}^N f_t(x_{0:t}^{(i)})w(x_{0:t}^{(i)})}{\frac{1}{N}\sum_{j=1}^N w(x_{0:t}^{(j)})} = \sum_{i=1}^N f_t(x_{0:t}^{(i)})\tilde{w}_t^{(i)}$$

$$\tilde{w}_t^{(i)} = \frac{w(x_{0:t}^{(i)})}{\sum_{j=1}^N w(x_{0:t}^{(j)})} \text{ : normalized importance weights}$$

$\hat{I}_N(f_t)$ converges to $I(f_t)$ as $N \to \infty$ with probability 1 by law of large numbers. It can be shown that under additional assumptions Central Limit Theorem guarantees that the rate of convergence is independent of dimension of the integrand, making this approach applicable to high-dimensions. We can also obtain the following approximation to $p(x_{0:t}|y_{1:t})$:

$$\hat{P}_N(dx_{0:t}|y_{1:t}) = \sum_{i=1}^N \tilde{w}_t^{(i)}\delta_{x_{0:t}^{(i)}}(dx_{0:t})$$

The above is useful, since we can sample from $\hat{P}_N(dx_{0:t}|y_{1:t})$, as long as we can easily sample from the proposal distribution π.

To avoid re-computing importance weights over the entire sequence, we can obtain an iterative version of the above approach. The estimate $\hat{P}_N(dx_{0:t}|y_{1:t})$ of $p(x_{0:t}|y_{1:t})$ can be computed incrementally without modifying $\{x_{0:t-1}^{(i)}, i = 1, ..., N\}$, and importance weights can be computed recursively:

$$\pi(x_{0:t}|y_{1:t}) = \pi(x_{0:t-1}|y_{1:t-1})\pi(x_t|x_{0:t-1}, y_{1:t}) = \pi(x_0)\prod_{k=1}^t \pi(x_k|x_{0:k-1}, y_{1:k}) \tag{10}$$

$$\tilde{w}_t^{(i)} \propto \tilde{w}_{t-1}^{(i)}\frac{p(y_t|x_t^{(i)})p(x_t^{(i)}|x_{t-1}^{(i)})}{\pi(x_t^{(i)}|x_{0:t-1}^{(i)}, y_{1:t})} \tag{11}$$

4.3 The Bootstrap Filter

The Bootstrap Filter is a basic algorithm from the family of Sequential Monte Carlo (SMC) algorithms. It is also referred to as SISR: sequential importance sampling with resampling. After sequential importance sampling we eliminate the particles having low importance weights and introduce a resampling step.

We use the prior distribution $p(x_{0:t}) = p(x_0)\prod_{k=1}^{t} p(x_k|x_{k-1})$ as importance distribution $\pi(x_{0:t}|y_{1:t})$. For resampling: we replace the weighted empirical distribution $\hat{P}_N(dx_{0:t}|y_{1:t}) = \sum_{i=1}^{N} \tilde{w}_t^{(i)} \delta_{x_{0:1}^{(i)}}(dx_{0:t})$ by the unweighted measure:

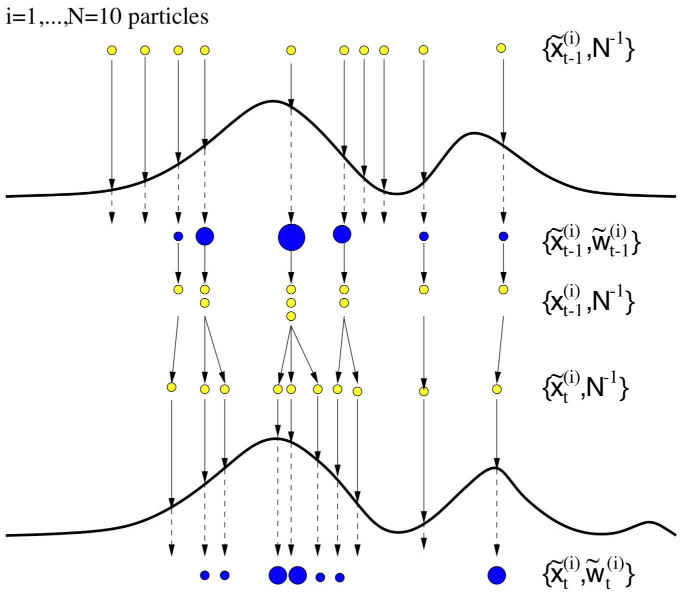
$$P_N(dx_{0:t}|y_{1:t}) = \frac{1}{N}\sum_{i=1}^{N} N_t^{(i)} \delta_{x_{0:1}^{(i)}}(dx_{0:t})$$

where $N_t^{(i)}$ is the number of offsprings associated with particle $x_{0:t}^{(i)}$ and $\sum_{i=1}^{N} N_t^{(i)} = N$. $N_t^{(i)}$ are constructed such that $P_N(dx_{0:t}|y_{1:t})$ is close to $\hat{P}_N(dx_{0:t}|y_{1:t})$ in the sense that for any f_t:

$$\int f_t(x_{0:t}) P_N(dx_{0:t}|y_{1:t}) \approx \int f_t(x_{0:t}) \hat{P}_N dx_{0:t}|y_{1:t}$$

For example, $N_t^{(i)}$ can be obtained by sampling N times from $\hat{P}_N(dx_{0:t}|y_{1:t})$. Figure 4 gives the details.

```
**Init:** For $i=1,...,N$, sample $x_0^{(i)} \sim p(x_0)$
      Set $t \leftarrow 1$
**Repeat:**
  **IS step:**
    For $i=1,...,N$, sample $\tilde{x}_t^{(i)} \sim p(x_t|x_{t-1}^{(i)})$
    Set $\tilde{x}_{0:t}^{(i)} \leftarrow (x_{0:t-1}^{(i)}, \tilde{x}_t^{(i)})$
    For $i=1,...,N$, evaluate $\tilde{w}_t^{(i)} = p(y_t|\tilde{x}_t^{(i)})$
  **Selection step:**
    Resample $N$ particles $\{x_{0:t}^{(i)} \mid i=1,...,N\}$
        from the set $\{\tilde{x}_{0:t}^{(i)} \mid i=1,...,N\}$
        using $\tilde{w}_t^{(i)}$ (with replacement)
  Set $t \leftarrow t+1$
```



i=1,...,N=10 particles
$\{\tilde{x}_{t-1}^{(i)}, N^{-1}\}$
$\{\tilde{x}_{t-1}^{(i)}, \tilde{w}_{t-1}^{(i)}\}$
$\{x_{t-1}^{(i)}, N^{-1}\}$
$\{\tilde{x}_t^{(i)}, N^{-1}\}$
$\{\tilde{x}_t^{(i)}, \tilde{w}_t^{(i)}\}$

Figure 5: (Left): Steps of the Bootstrap algorithm. (Right): A graphical representation from [3]: the bootstrap filter starts at time t-1 with an unweighted measure $\{\tilde{x}_{t-1}^{(i)}, 1/N\} \approx p(x_{t-1}|y_{1:t-2})$; for each particle we compute $\tilde{w}_{t-1}^{(i)}$, yielding the weighted measure $\{\tilde{x}_{t-1}^{(i)}, \tilde{w}_{t-1}^{(i)}\} \approx p(x_{t-1}|y_{1:t-1})$; the resampling step selects the fittest particles to get the unweighted measure $\{\tilde{x}_{t-1}^{(i)}, 1/N\} \approx p(x_{t-1}|y_{1:t-1})$; finally, the next sampling (prediction) step introduces variety, resulting in the measure $\{\tilde{x}_t^{(i)}, 1/N\} \approx p(x_t|y_{1:t-1})$. Note that $\tilde{w}_{t-1}$ does not appear when computing $\tilde{w}_t^{(i)} = p(y_t|\tilde{x}_t^{(i)})$, because the propagated particles $x_{0:t-1}^{(i)}$ have uniform weights after the resampling step.

4.4 Exercises

4.4.1 Analytic Exercise [1pt]

Show that $\tilde{w}_t^i \propto \tilde{w}_{t-1}^i \frac{p(y_t|x_t^{(i)})p(x_t^{(i)}|x_{t-1}^{(i)})}{\pi(x_t^{(i)}|x_{0:t-1}^{(i)}, y_{1:t})}$ (Equation 11 from Section 4.2).

4.4.2 Coding Exercise [4pts]

Suppose we have the following model:

$$x_t = \tfrac{1}{2}x_{t-1} + 25\frac{x_{t-1}}{1+x_{t-1}^2} + 8\cos(1.2t) + v_t$$

$$y_t = \frac{x_t^2}{20} + w_t$$

where $x_1 \sim \mathcal{N}(0, \sigma_1^2 = 10)$, $v_t \sim \mathcal{N}(0, \sigma_v^2 = 10)$ and $w_t \sim \mathcal{N}(0, \sigma_w^2 = 1)$ are Gaussian noise terms.

Implement the Bootstrap filter (in any language of your choice) and plot the estimated filtering distribution $p(x_t|y_{1:t})$ on a 2D grid of $x_t \in [-25, 25]$ and $t \in [0, 200]$. Figure 5 shows an example plot for $t \in [0, 100]$ using 1000 particles.

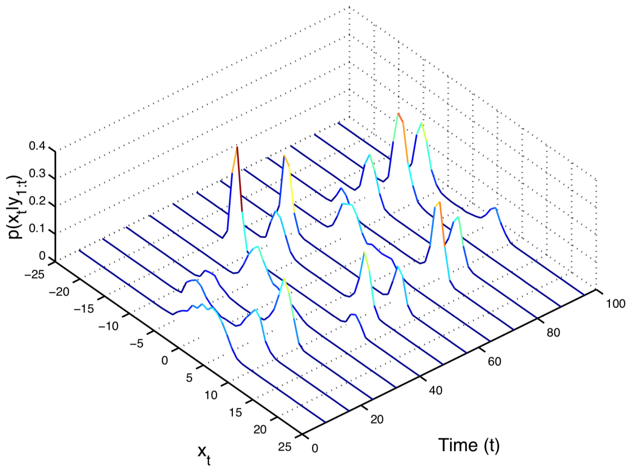


Figure 6: Estimated filtering distribution plot from [3].

Change the number of particles and determine:

1. the minimum number of particles needed for multimodality to emerge quickly in this example
2. can you find N for this problem s.t. increasing the number of particles does not change the appearance of the filtering distribution on your plot?

5 Summary and Further Reading

In this tutorial we introduced the basics of MCMC methods. A brief summary of MCMC approaches used in Machine Learning can be found in [1]. A more in-depth treatment can be found in [4]. These include discussions of Hybrid MC methods (that combine several MCMC kernels) and physics-inspired approaches, for example that Hamiltonian Monte Carlo that uses Hamiltonian dynamics.

For settings with sequential data, this tutorial introduced a basic Sequential Monte Carlo algorithm – the Bootstrap Filter. [3] and [5] provide further information on SMC theory and examples of SMC algorithms.

References

[1] Christophe Andrieu, Nando De Freitas, Arnaud Doucet, and Michael I Jordan. An Introduction to MCMC for Machine Learning. *Machine learning*, 50(1-2):5–43, 2003.

[2] Particle filter. *https://en.wikipedia.org/wiki/Particle_filter.*

[3] Arnaud Doucet, Nando de Freitas, and Neil Gordon. *Sequential Monte Carlo Methods in Practice.* Springer, 2001.

[4] Steve Brooks, Andrew Gelman, Galin Jones, and Xiao-Li Meng. *Handbook of Markov Chain Monte Carlo.* CRC press, 2011.

[5] Thomas B. Schön, Fredrik Lindsten, Johan Dahlin, Johan Wagberg, Christian A. Naesseth, Andreas Svensson, and Liang Dai. Sequential Monte Carlo methods for system identification. *17th IFAC Symposium on System Identification (SYSID)*, 2015.