Markov Chain Monte Carlo
Theory and Practical applications
Session 4 : Some practical considerations

Randal Douc and Sylvain Le Corff

Télécom SudParis, Institut Polytechnique de Paris
sylvain.le_corff@telecom-sudparis.eu

Institut Mines-Télécom

Useful books

- Feynman-Kac formulae: genealogical and interacting particle systems with applications,
  Del Moral P., 2004, Springer.

- Inference in hidden Markov models,
  Cappé O., Moulines E. and Rydén T., 2005, Springer.

- Nonlinear time series: theory, methods and applications with R examples,
  Douc R., Moulines E. and Stoffer D., 2014, Chapman & Hall.

Some examples and illustrations are borrowed from:

Nonlinear time series: theory, methods and applications with R examples,
Douc R., Moulines E. and Stoffer D., 2014, Chapman & Hall.

R codes available at
http://www.stat.pitt.edu/stoffer/nltsa/Rcode.html.

Bayesian setting

In a Bayesian setting, a parameter X is embedded with a prior distribution p and the observations are given by a probabilistic model:

$$Y \sim \ell(\cdot|X) \, .$$

The inference is then based on the posterior distribution:

$$\boxed{\pi(x|Y) = \frac{p(x)\ell(Y|x)}{\int p(u)\ell(Y|u)\mathrm{d}u} \, .}$$

In most cases the normalizing constant is not tractable:

$$\pi(X|Y) \propto p(X)\ell(Y|X) \, .$$

Binary regression set-up

The observations $(Y_1, \ldots, Y_n)$ are conditionally independent Bernoulli random variables with success probability $F(X^T z_i)$.

- z_i d-dimensional vector of known covariates.
- X unknown regression coefficient.
- F known distribution function (normal, logistic).

Prior distribution of X:

$$\log \pi_0(X) \sim -X^T \Sigma^{-1} X/2 \quad \text{or} \quad \log \pi_0(X) \sim -\lambda \|X\|_1 \,.$$

Posterior distribution of X:

$$\pi\left(X|Y\right) \propto \exp\left\{-U(X)\right\} \,,$$

$$U(X) = -\sum_{i=1}^{n} Y_i \log\left\{F(X^T z_i)\right\} + (1 - Y_i) \log\left\{1 - F(X^T z_i)\right\} + \pi_0(X) \,.$$

Bayesian setting

Bayesian decision theory relies on minimization problems involving expectations:

$$\int L(x, \theta) p(x) \ell(Y|x) \mathrm{d}x \ .$$

Generic problem: estimation of an expectation $\mathbb{E}_\pi[f]$ where

- π is known up to a multiplicative factor;
- we do not know how to sample from π (no basic Monte Carlo estimator);
- π is high dimensional density (usual importance sampling and accept/reject inefficient).

MCMC: rationale

Let X_1 be any starting point.

- For a given target distribution π, choose a π-reversible transition kernel with density k:

$$\pi(x)k(x,x') = \pi(x')k(x',x) \quad \text{[Reversibility]}$$

- Sample a Markov chain $X_1, \ldots, X_n$ with kernel k and compute

$$\hat{\pi}_n(f) = \frac{1}{n}\sum_{i=1}^{n} f(X_i)$$

  to approximate $\pi(f) = \int f(x)\pi(\mathrm{d}x)$.

$\Rightarrow$ Doest it converge ? What is the rate of convergence ?

MCMC: rationale

Under regularity assumptions, if π is a stationary distribution:

1. **Ergodic theorem**: [Chapter 3], under which condition can we establish, for $f \in \mathrm{L}^1(\pi)$,

$$\hat{\pi}(f) = \frac{1}{n}\sum_{i=1}^{n} f(X_i) \xrightarrow[a.s.]{} \int f(x)\pi(x)\mathrm{d}x\,.$$

2. **Central limit theorem**: [Chapter 4], under which condition can we establish, for $f \in \mathrm{L}^1(\pi)$,

$$\frac{\sqrt{n}}{\sigma_{\pi,q,f}}\left[\frac{1}{n}\sum_{i=1}^{n} f(X_i) - \int f(x)\pi(x)\mathrm{d}x\right] \xrightarrow{\mathcal{D}} \mathcal{N}(0,1)\,.$$

Key tool : the Accept-Reject algorithm

Assume we know that $\pi(x) \leq Mr(x)$ and that we know how to sample from r.

1. Sample $X \sim r$ and $U \sim U([0,1])$.
2. If
$$U \leq \frac{\pi(X)}{Mr(X)},$$
accept X.
3. Else go to 1.

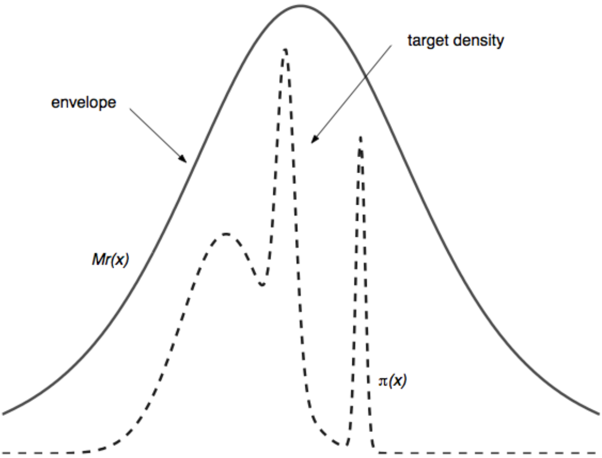

Figure: Illustration of the Accept-Reject method (Cappé, Moulines, Ryden 2005).

The Metropolis-Hastings algorithm

- Objective target density π.
- Instrumental transition density $q(x, y)$.

Given X_k,

1. Generate $Y_{k+1} \sim q(\cdot, X_k)$.
2. Set

$$X_{k+1} = \begin{cases} Y_{k+1} & \text{with probability } \alpha(X_k, Y_{k+1})\,, \\ X_k & \text{with probability } 1 - \alpha(X_k, Y_{k+1})\,. \end{cases}$$

where

$$\boxed{\alpha(x, y) = 1 \wedge \frac{\pi(y)}{\pi(x)} \frac{q(y, x)}{q(x, y)}}\,.$$

$\Rightarrow$ No restriction on π and q, with this choice of α the algorithm produces a Markov chain with stationary distribution π.

The Metropolis-Hastings algorithm

```python
def HM_monte_carlo(n_samples, log_prob, initial_state, step_size = 0.1):
    """
    Inputs
    ----------
    n_samples: number of samples to return
    log_prob: opposite of the loglikelihood to sample from
    initial_state: initial sample
    step_size: standard deviation of the proposed moves

    Outputs
    -------
    samples: samples from the MCMC algorithm
    accepted: array of 0 and 1 to display which proposed moves have been accepted
    """
```

The Metropolis-Hastings algorithm

```
def HM_monte_carlo(n_samples, log_prob, initial_state, step_size = 0.1):

    initial_state = np.array(initial_state)

    samples  = [initial_state]
    accepted = []

    size = (n_samples,) + initial_state.shape[:1]

    # random variable to sample proposed moves
    epsilon = st.norm(0, 1).rvs(size)

    for noise in tqdm(epsilon):

        q_new = samples[-1] + step_size*noise

        # acceptance rate
        old_log_p = log_prob(samples[-1])
        new_log_p = log_prob(q_new)

        if np.log(np.random.rand()) < old_log_p - new_log_p:
            samples.append(q_new)
            accepted.append(True)
        else:
            samples.append(np.copy(samples[-1]))
            accepted.append(False)

    return (np.array(samples[1:]),np.array(accepted),)
```

Independent case

In this case $q(x,y) = g(y)$.

1. Generate $Y_{k+1} \sim g(\cdot)$.
2. Set

$$X_{k+1} = \begin{cases} Y_{k+1} & \text{with probability } \alpha(X_k, Y_{k+1})\,, \\ X_k & \text{with probability } 1-\alpha(X_k, Y_{k+1})\,. \end{cases}$$

where

$$\boxed{\alpha(x,y) = 1 \wedge \frac{\pi(y)}{\pi(x)} \frac{g(x)}{g(y)}}\,.$$

Alternative to importance sampling and Accept-Reject algorithms.

Independent case

The samples are not i.i.d. but, if there exists M such that $\pi(x) \leqslant Mg(x)$ then

$$\|K^n(x,\cdot) - \pi\|_{tv} \leqslant \left(1 - \frac{1}{M}\right)^n \quad \text{(Ergodicity)}$$

(Roberts, Tweedie 1996), (Mengersen, Tweedie 1996).

Expected acceptance probability is $1/M$, no need to know M.

If the majoration condition does not hold, no geometric ergodicity.

Cauchy vs Normal (I)

- Target distribution: $\pi(x) \propto (1 + x^2)^{-1}$.
- Proposal distribution: $g(y) \sim \mathcal{N}(0, 1)$.

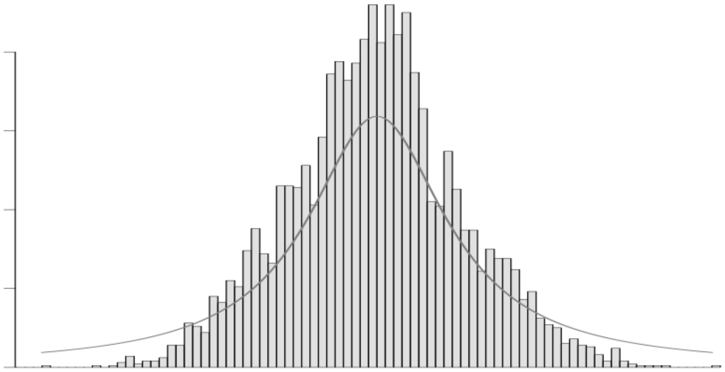

Figure: Histogram of IMH with 5000 samples.

Random walk Metropolis-Hastings

The proposal mechanism is given by $Y_{k+1} = X_k + \varepsilon_{k+1}$, where ε_{k+1} is independent of X_{k+1}. The proposal distribution is of the form $q(x, y) = q(y - x)$ with q is symmetric.

1. Generate $Y_{k+1} \sim q(X_k, \cdot)$.
2. Set

$$X_{k+1} = \begin{cases} Y_{k+1} & \text{with probability } \alpha(X_k, Y_{k+1})\,, \\ X_k & \text{with probability } 1 - \alpha(X_k, Y_{k+1})\,. \end{cases}$$

where

$$\boxed{\alpha(x, y) = 1 \wedge \frac{\pi(y)}{\pi(x)}}\,.$$

Using random walk moves prevents from being uniformly ergodic (Robert, Casella 2004).
But still, geometric ergodicity.

Cauchy vs Normal (II)

- Target distribution: $\pi(x) \propto (1 + x^2)^{-1}$.
- Proposal distribution: $\mathcal{N}(0, 1)$.

$$\alpha(x, y) = 1 \wedge \frac{1 + x^2}{1 + y^2}$$

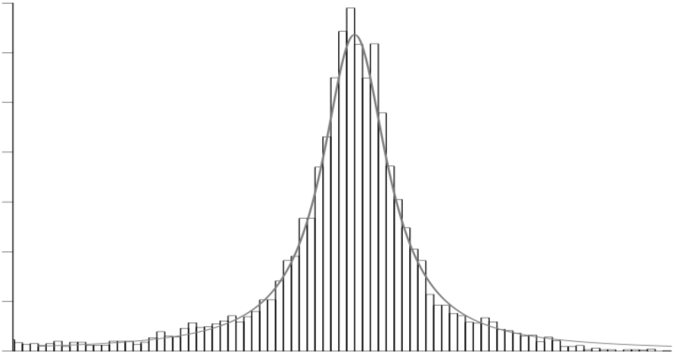

Figure: Histogram of IMH with 10000 samples.

Improving Metropolis-Hastings using gradient information

Langevin equation associated with π:

$$\boxed{\mathrm{d}X_t = (\Sigma/2)\nabla \log \pi(X_t)\mathrm{d}t + \Sigma^{1/2}\mathrm{d}W_t \ ,}$$

where W is a d-dimensional Brownian motion.

Under appropriate regularity assumptions, the generated dynamic is ergodic with unique invariant distribution π.

Solving this equation analytically would allow to sample exacty from π. Not tractable in practice!

Another family of proposals is based on the Euler-Maruyama discretization of the equation...
Proposal mechanism of the form

$$Y_{k+1} = X_k + \frac{h\sigma^2}{2}\nabla \log \pi(X_k) + \sqrt{h}\sigma\varepsilon_{k+1}\,.$$

The MALA algorithm

```python
def MALA_monte_carlo(n_samples, log_prob, initial_state, step_size = 0.1):
    """
    Inputs
    ----------
    n_samples: number of samples to return
    log_prob: opposite of the loglikelihood to sample from
    initial_state: initial sample
    step_size: standard deviation of the proposed moves

    Outputs
    -------
    samples: samples from the MCMC algorithm
    accepted: array of 0 and 1 to display which proposed moves have been accepted
```

The MALA algorithm

```
def MALA_monte_carlo(n_samples, log_prob, initial_state, step_size = 0.1):

    initial_state = np.array(initial_state)

    gradV = grad(log_prob)

    samples  = [initial_state]
    accepted = []

    size = (n_samples,) + initial_state.shape[:1]

    # random variable to sample proposed moves
    epsilon = st.norm(0, 1).rvs(size)
    step    = 0.5/(step_size**2)
    for noise in tqdm(epsilon):

        grad_new = gradV(samples[-1])
        mean_new = samples[-1] - step*grad_new
        q_new    = mean_new + step_size*noise

        grad_y   = gradV(q_new)
        mean_y   = q_new - step*grad_y

        # acceptance rate
        old_log_p = log_prob(samples[-1]) + step*np.dot(q_new-mean_new,q_new-mean_new)
        new_log_p = log_prob(q_new) + step*np.dot(samples[-1]-mean_y,samples[-1]-mean_y)

        if np.log(np.random.rand()) < old_log_p - new_log_p:
            samples.append(q_new)
            accepted.append(True)
        else:
            samples.append(np.copy(samples[-1]))
            accepted.append(False)

    return (np.array(samples[1:]),np.array(accepted),)
```

Challenge I: scaling issues and high dimensionality

How to **choose the scaling (σ)** of the algorithm to optimize efficiency ?

Scaling problem mainly studied for:

1. Random walk Metropolis-Hastings (RWM)
   - Proposal mechanism of the form $Y_{k+1} = X_k + \sigma \varepsilon_{k+1}$.
   - Acceptance rate:

$$\boxed{\alpha(x, y) = 1 \wedge \frac{\pi(y)}{\pi(x)}}.$$

2. Metropolis-Adjusted Langevin Algorithm (MALA)
   - Proposal mechanism of the form

$$Y_{k+1} = X_k + \frac{h\sigma^2}{2} \nabla \log \pi(X_k) + \sqrt{h} \sigma \varepsilon_{k+1} \, .$$