

Machine Learning Summer School 2003



Practical Session: Simulation methods

Manuel Davy

1 Introduction

Most elements of this practical session are inspired from the paper by C. Andrieu and A. Doucet, *Joint Bayesian Model Selection and Estimation of Noisy Sinusoids via Reversible Jump MCMC*, IEEE Transactions on Signal Processing, Vol. 47, No 9, October 1999.

1.1 A linear model

Let y_t , $t = 1, \dots, N$, be a time series, where t denotes the time. We denote in vector form $\mathbf{y} = [y_1, \dots, y_N]^T$. We assume that \mathbf{y} is modelled as follows:

$$\mathbf{y} = \mathbf{D}(\boldsymbol{\theta})\boldsymbol{\beta} + \boldsymbol{\epsilon} \quad (1)$$

where

- $\mathbf{D}(\boldsymbol{\theta})$ is a $N \times L$ matrix whose columns contain basis functions (examples are given below). The vector $\boldsymbol{\theta}$ is composed of possible parameters of \mathbf{D} ,
- $\boldsymbol{\beta}$ is a vector of amplitude coefficients, with size L ,
- $\boldsymbol{\epsilon} = [\epsilon_1, \dots, \epsilon_N]^T$ is the noise vector where ϵ_t , $t = 1, \dots, N$ are i.i.d. zero-mean Gaussian noise samples with variance σ_ϵ^2 .

Example 1: Spectral estimation. A common Signal Processing problem consists of estimating the frequencies of sinusoids embedded in noise (*Spectral estimation*). The corresponding model is:

$$\text{for } t = 1, \dots, N, \quad y_t = \sum_{i=1}^I \beta_{i,c} \cos(\omega_i t) + \beta_{i,s} \sin(\omega_i t) + \epsilon_t \quad (2)$$

In this case, the parameter vector $\boldsymbol{\theta}$ is $\boldsymbol{\theta} = [I; \boldsymbol{\omega}]$ with $\boldsymbol{\omega} = [\omega_1, \dots, \omega_I]^T$ and

$$\mathbf{D}(\boldsymbol{\theta}) = \begin{bmatrix} \cos(\omega_1) & \sin(\omega_1) & \cdots & \cos(\omega_I) & \sin(\omega_I) \\ \cos(\omega_1 2) & \sin(\omega_1 2) & \cdots & \cos(\omega_I 2) & \sin(\omega_I 2) \\ \cos(\omega_1 3) & \sin(\omega_1 3) & \cdots & \cos(\omega_I 3) & \sin(\omega_I 3) \\ \vdots & \vdots & \cdots & \vdots & \vdots \\ \cos(\omega_1 N) & \sin(\omega_1 N) & \cdots & \cos(\omega_I N) & \sin(\omega_I N) \end{bmatrix} \begin{matrix} (t=1) \\ (t=2) \\ (t=3) \\ (t=N) \end{matrix}. \quad (3)$$

The amplitude vector $\boldsymbol{\beta}$ is

$$\boldsymbol{\beta} = [\beta_{1,c}, \beta_{1,s}, \dots, \beta_{I,c}, \beta_{I,s}]^T \quad (4)$$

and $L = 2I$. \square

Example 2: Blind deconvolution. Another generic problem is that of blind deconvolution. In this case,

$$\text{for } t = 1, \dots, N, \quad y_t = \sum_{i=1}^I \beta_i h(t - t_i) + \epsilon_t \quad (5)$$

where h is an unknown impulse response. The corresponding matrix $\mathbf{D}(\boldsymbol{\theta})$ is (with $L = I$):

$$\mathbf{D}(\boldsymbol{\theta}) = \begin{bmatrix} h(1 - t_1) & \cdots & h(1 - t_I) \\ h(2 - t_1) & \cdots & h(2 - t_I) \\ h(3 - t_1) & \cdots & h(3 - t_I) \\ \vdots & \cdots & \vdots \\ h(N - t_1) & \cdots & h(N - t_I) \end{bmatrix} \begin{matrix} (t=1) \\ (t=2) \\ (t=3) \\ (t=N) \end{matrix}. \quad (6)$$

and the parameter vector is $\boldsymbol{\theta} = [I, h(-N + 1), \dots, h(N)]$. \square

1.2 Likelihood, prior and posterior

The likelihood of \mathbf{y} is completely defined by the elements given above:

$$p(\mathbf{y}|\boldsymbol{\theta}, \boldsymbol{\beta}, \sigma_\epsilon^2) = \frac{1}{(2\pi\sigma_\epsilon^2)^{N/2}} \exp \left\{ -\frac{1}{2\sigma_\epsilon^2} [\mathbf{y} - \mathbf{D}(\boldsymbol{\theta})\boldsymbol{\beta}]^T [\mathbf{y} - \mathbf{D}(\boldsymbol{\theta})\boldsymbol{\beta}] \right\} \quad (7)$$

One generally considers a zero-mean Gaussian amplitude prior, with covariance matrix $\sigma_\epsilon^2 \boldsymbol{\Sigma}_\beta$, i.e.

$$p(\boldsymbol{\beta}|\sigma_\epsilon^2) = \frac{1}{(2\pi\sigma_\epsilon^2)^{L/2} \sqrt{\det(\boldsymbol{\Sigma}_\beta)}} \exp \left(-\frac{1}{2\sigma_\epsilon^2} \boldsymbol{\beta}^T \boldsymbol{\Sigma}_\beta^{-1} \boldsymbol{\beta} \right) \quad (8)$$

From Bayes rule, the posterior distribution of $(\boldsymbol{\theta}, \boldsymbol{\beta}, \sigma_\epsilon^2)$ is given by $p(\boldsymbol{\theta}, \boldsymbol{\beta}, \sigma_\epsilon^2|\mathbf{y}) \propto p(\mathbf{y}|\boldsymbol{\theta}, \boldsymbol{\beta}, \sigma_\epsilon^2) p(\boldsymbol{\beta}|\sigma_\epsilon^2, \boldsymbol{\theta}) p(\sigma_\epsilon^2) p(\boldsymbol{\theta})$ (up to a multiplicative factor):

$$\begin{aligned} p(\boldsymbol{\theta}, \boldsymbol{\beta}, \sigma_\epsilon^2|\mathbf{y}) &\propto \frac{1}{(2\pi\sigma_\epsilon^2)^{(L+N)/2} \sqrt{\det(\boldsymbol{\Sigma}_\beta)}} \exp \left\{ -\frac{1}{2\sigma_\epsilon^2} [\boldsymbol{\beta} - \boldsymbol{\mu}]^T \mathbf{S}^{-1} [\boldsymbol{\beta} - \boldsymbol{\mu}] \right\} \\ &\times \exp \left[-\frac{1}{2\sigma_\epsilon^2} (\boldsymbol{\mu}^T \mathbf{S}^{-1} \boldsymbol{\mu} + \mathbf{y}^T \mathbf{y}) \right] p(\sigma_\epsilon^2) p(\boldsymbol{\theta}) \end{aligned} \quad (9)$$

where \mathbf{S} is a $L \times L$ matrix given by

$$\mathbf{S} = [\mathbf{D}(\boldsymbol{\theta})^T \mathbf{D}(\boldsymbol{\theta}) + \boldsymbol{\Sigma}_{\boldsymbol{\beta}}^{-1}]^{-1} \quad (10)$$

and $\boldsymbol{\mu}$ is a vector of size L :

$$\boldsymbol{\mu} = \mathbf{S} \mathbf{D}(\boldsymbol{\theta})^T \mathbf{y} \quad (11)$$

1.3 Marginal posterior

In Eq. (9), it is possible to integrate out the amplitudes vector $\boldsymbol{\beta}$, since it is a Gaussian distribution with mean $\boldsymbol{\mu}$ and covariance matrix \mathbf{S} . This yields

$$p(\boldsymbol{\theta}, \sigma_{\epsilon}^2 | \mathbf{y}) \propto \frac{1}{(2\pi\sigma_{\epsilon}^2)^{N/2}} \sqrt{\frac{\det(\mathbf{S})}{\det(\boldsymbol{\Sigma}_{\boldsymbol{\beta}})}} \exp \left[-\frac{1}{2\sigma_{\epsilon}^2} (\mathbf{y}^T \mathbf{y} - \boldsymbol{\mu}^T \mathbf{S}^{-1} \boldsymbol{\mu}) \right] p(\sigma_{\epsilon}^2) p(\boldsymbol{\theta}) \quad (12)$$

Additional marginalisation is possible if one chooses, e.g., $\boldsymbol{\Sigma}_{\boldsymbol{\beta}}$ to be the *g-prior*, namely

$$\boldsymbol{\Sigma}_{\boldsymbol{\beta}}^{-1} = \frac{1}{\xi^2} \mathbf{D}(\boldsymbol{\theta})^T \mathbf{D}(\boldsymbol{\theta})$$

and select a conjugate prior for σ_{ϵ}^2 , e.g., an inverse Gamma distribution $p(\sigma_{\epsilon}^2) = \mathcal{IG}(\sigma_{\epsilon}^2; \nu_0, \nu_1)$. When σ_{ϵ}^2 is integrated out, the following result holds

$$p(\boldsymbol{\theta} | \mathbf{y}) \propto (\mathbf{y}^T \mathbf{P} \mathbf{y} + 2\nu_1)^{-N/2 - \nu_0} \boldsymbol{\theta} \quad (13)$$

with

$$\mathbf{P} = \mathbf{I}_N - \mathbf{D}(\boldsymbol{\theta}) \mathbf{S} \mathbf{D}(\boldsymbol{\theta})^T \quad (14)$$

where \mathbf{I}_N is the identity matrix of size N . We also have the conditional distributions

$$p(\sigma^2 | \boldsymbol{\theta}, \mathbf{y}) = \mathcal{IG}(\sigma^2; N/2 + \nu_0, \mathbf{y}^T \mathbf{P} \mathbf{y} / 2 + \nu_1) \quad (15)$$

$$p(\boldsymbol{\beta} | \sigma^2, \boldsymbol{\theta}, \mathbf{y}) = \mathcal{N}(\boldsymbol{\beta}; \boldsymbol{\mu}, \sigma^2 \mathbf{S}) \quad (16)$$

2 MCMC sampling (work to be done)

Consider the frequency estimation problem (Example 1). We choose a uniform prior for the frequencies, i.e., $p(\boldsymbol{\theta}) = \pi^{-I}$. We want to sample from the frequencies posterior, i.e., from $p(\boldsymbol{\omega} | \mathbf{y})$ for, e.g., frequency estimation. In order to do so, we will implement an MCMC algorithm.

Some MCMC theory. MCMC stands for *Monte Carlo Markov Chain*. The Monte Carlo part consists of approximating an Expectation by a mean. More precisely, assume we want to compute

$$\mathbf{E}[h(\boldsymbol{\theta})] = \int h(\boldsymbol{\theta}) p(\boldsymbol{\theta} | \mathbf{y}) d\boldsymbol{\theta} \quad (17)$$

then a good approximation is given by

$$\mathbf{E}[h(\boldsymbol{\theta})] \approx \frac{1}{L} \sum_{l=1}^L h(\tilde{\boldsymbol{\theta}}^l) \quad (18)$$

where $\tilde{\boldsymbol{\theta}}^l$ is a set of samples distributed according to $p(\boldsymbol{\theta}|\mathbf{y})$. The Markov chain part consists of building a Markov chain $\{\tilde{\boldsymbol{\theta}}^1, \tilde{\boldsymbol{\theta}}^2, \tilde{\boldsymbol{\theta}}^3, \dots\}$ with invariant distribution $p(\boldsymbol{\theta}|\mathbf{y})$. A Markov chain is completely defined (in terms of its distribution) by the initial distribution of $\tilde{\boldsymbol{\theta}}^1$ and by a so-called *transition kernel* $K(\tilde{\boldsymbol{\theta}}^l, d\boldsymbol{\theta}^{l+1})$. Standard algorithms such as Metropolis-Hastings, or the Gibbs sampler, define transition kernels that ensure convergence to the target distribution $p(\boldsymbol{\theta}|\mathbf{y})$, whatever the initial distribution of $\tilde{\boldsymbol{\theta}}^1$. Note however, that the speed of convergence can be quite slow and careful kernel design is necessary in practice.

The Metropolis-Hastings Algorithm. This algorithm defines an MCMC kernel with a given invariant distribution. Here, the invariant distribution we consider is $p(\boldsymbol{\theta}|\mathbf{y})$. This algorithm works in two steps:

1. From the previous accepted sample, denoted $\tilde{\boldsymbol{\theta}}^l$, sample a *candidate* $\boldsymbol{\theta}^*$ by using a *proposal distribution* $q(\boldsymbol{\theta}^*|\tilde{\boldsymbol{\theta}}^l)$
2. Accept the candidate with probability α (i.e., set $\tilde{\boldsymbol{\theta}}^{l+1} = \boldsymbol{\theta}^*$) and with probability $1 - \alpha$, reject it (i.e., set $\tilde{\boldsymbol{\theta}}^{l+1} = \tilde{\boldsymbol{\theta}}^l$)

The probability α is defined as follows

$$\alpha = \min \left\{ 1, \frac{p(\boldsymbol{\theta}^*|\mathbf{y})}{p(\tilde{\boldsymbol{\theta}}^l|\mathbf{y})} \frac{q(\tilde{\boldsymbol{\theta}}^l|\boldsymbol{\theta}^*)}{q(\boldsymbol{\theta}^*|\tilde{\boldsymbol{\theta}}^l)} \right\} \quad (19)$$

Now, let's see how it works in practice.

Note. Matlab functions that may be necessary are listed in appendix and corresponding files are provided at the start of the session. Pseudo code is also reported in Appendix A, but it is recommended to refer to it for verification of your own codes only.

2.1 A first basic Metropolis-Hastings algorithm: One-at-a-time independent MH

In order to implement a basic Metropolis-Hastings algorithms, we are going through the following steps.

1. Write the pseudo-code of a one-at-a-time Metropolis-Hastings algorithm (see Appendix A.1 for verification)
2. Select an independent proposal distribution¹ $q_1(\omega_i)$, for any $i = 1, \dots, I$. One can use a uniform proposal or, more efficiently, the distribution defined as a stepwise approximation of the magnitude of the Fourier transform of \mathbf{y} . Write the MH ratio and implement the algorithm (matlab files provided may be used).

¹A proposal distribution is said *independent* whenever it does not depend on the last element of the Markov chain.

3. Test the algorithm with a time series of $N = 100$ samples and $I = 3$ frequencies. Can you identify the burn-in phase? Does the chain distribution seem to converge to what you expect (plot the histogram)? How could we improve the algorithm?

2.2 A second basic Metropolis-Hastings algorithm: One-at-a-time random walk MH

A solution to overcome the problems met with the previous algorithm consists of replacing $q_1(\omega_i)$, $i = 1, \dots, I$ by a random walk proposal distribution, that depends on previous element of the Markov chain, denoted $q_2(\omega_i|\tilde{\omega}_i^l)$. More precisely, we choose

$$q_2(\omega_i|\tilde{\omega}_i^l) = \mathcal{N}(\omega_i; \tilde{\omega}_i^l, \sigma_{\text{RW}}^2) \quad (20)$$

1. Implement this algorithm (do not delete the previous one !)
2. Compare its performance to those of the previous algorithm (plot histograms);
3. Any conclusions?

2.3 A third Metropolis-Hastings algorithm: One-at-a-time hybrid MH

We propose now to mix the previous two algorithms. The idea is as follows: with probability λ , select q_1 as proposal distribution, and with probability $1 - \lambda$, choose q_2 as proposal distribution.

1. What is the corresponding Markov chain kernel ?
2. Implement this algorithm;
3. Study its performance and conclude;
4. Add sampling of the marginalised parameters σ^2 and β from their posterior distributions. Any conclusions concerning markov Chain convergence?

2.4 A Metropolis-within-Gibbs algorithm: hyperparameter sampling

Gibbs sampling is another way of sampling a Markov chain. Assume we want to sample from $p(a, b)$ where a and b are scalar parameters. Gibbs sampling consists of the following two steps, implemented at each iteration l :

1. sample $\tilde{a}_l \sim p(a|\tilde{b}_{l-1})$
2. sample $\tilde{b}_l \sim p(b|\tilde{a}^{l-1})$

Obviously, this requires to be able to sample directly from the conditional distributions $p(a|b)$ and $p(b|a)$, which might not be possible. It is still possible, though, to include a MH step into the Gibbs sampler, yielding the Metropolis-within-Gibbs algorithm.

1. Let us assume an Inverse Gamma prior for the hyperparameter ξ^2 . Calculate its conditional posterior $p(\xi^2|\omega, \sigma^2, \beta, \mathbf{y}) = p(\xi^2|\omega, \sigma^2, \beta)$.
2. Implement the Metropolis-within-Gibbs algorithm to sample $(\xi^2, \omega, \sigma^2, \beta)$.

A Algorithms

This appendix gives pseudo code of algorithms implemented in Subsections 2.1 to 2.4.

A.1 One-at-a-time, independent MH algorithm for spectral estimation

```
% Initialisation
- Set  $l \leftarrow 1$  %  $n$  is the Markov chain iteration number
- For  $i = 1$  to  $I$ , % Initialisation of the  $I$  frequencies
    * sample  $\tilde{\omega}_i^1 \sim q_1(\omega|\mathbf{y})$  %  $q_1$  is the proposal distribution, based on FT of  $\mathbf{y}$ 

% Iterations of the MH algorithm
- For  $l = 2$  to  $l_{\max}$ ,
    * For  $i = 1$  to  $I$ , % Note: this is a one-at-a-time algorithm
        . Set  $\omega^* \leftarrow \tilde{\omega}^{l-1}$ 
        . Sample  $\omega_i^* \sim q_1(\omega_i|\mathbf{y})$  % current component  $\#i$  only is modified
        . % Computation of the MH ratio
        . Compute  $p(\omega^*|\mathbf{y})$ 
        . Compute  $p(\tilde{\omega}^{l-1}|\mathbf{y})$ 
        . Compute  $q_1(\omega_i^*|\mathbf{y})$ 
        . Compute  $q_1(\tilde{\omega}_i^{l-1}|\mathbf{y})$ 
        . Finally, compute  $\alpha$  as given in Eq. (19).
        . % Accept/reject test
        . Sample  $u \sim \mathcal{U}[0; 1]$  %  $u$  is a uniform random variable on  $[0;1]$ 
        . If  $\alpha > u$ , Set  $\tilde{\omega}^l \leftarrow \omega^*$  % the candidate is accepted
        . Otherwise, Set  $\tilde{\omega}^l \leftarrow \tilde{\omega}^{l-1}$  % the candidate is accepted
```

A.2 One-at-a-time random walk MH

```
% Initialisation
- Set  $l \leftarrow 1$  %  $n$  is the Markov chain iteration number
- For  $i = 1$  to  $I$ , % Initialisation of the  $I$  frequencies
    * sample  $\tilde{\omega}_i^1 \sim \mathcal{U}(\omega; [0, \pi])$  %  $\mathcal{U}(\omega; [0, \pi])$  is the uniform distribution on  $[0, \pi]$ 

% Iterations of the MH algorithm
- For  $l = 2$  to  $l_{\max}$ ,
    * For  $i = 1$  to  $I$ , % Note: this is a one-at-a-time algorithm
        . Set  $\omega^* \leftarrow \tilde{\omega}^{l-1}$ 
        . Sample  $\omega_i^* \sim q_2(\omega_i|\tilde{\omega}_i^{l-1}) = \mathcal{N}(\omega; \tilde{\omega}_i^{l-1}, \sigma_{\text{RW}}^2)$ 
        . % Computation of the MH ratio
        . Compute  $p(\omega^*|\mathbf{y})$ 
        . Compute  $p(\tilde{\omega}^{l-1}|\mathbf{y})$ 
        . Finally, compute  $\alpha$  as given in Eq. (19), noting that  $q_2(\omega_i|\tilde{\omega}_i^{l-1}) = q_2(\tilde{\omega}^{l-1}|\omega)$ 
        . % Accept/reject test
        . Sample  $u \sim \mathcal{U}[0; 1]$  %  $u$  is a uniform random variable on  $[0;1]$ 
        . If  $\alpha > u$ , Set  $\tilde{\omega}^l \leftarrow \omega^*$  % the candidate is accepted
        . Otherwise, Set  $\tilde{\omega}^l \leftarrow \tilde{\omega}^{l-1}$  % the candidate is accepted
```

A.3 One-at-a-time hybrid MH

```
% Initialisation

– Choose  $\lambda$  in  $[0; 1]$ 

– Set  $l \leftarrow 1$  %  $n$  is the Markov chain iteration number

– For  $i = 1$  to  $I$ , % Initialisation of the  $I$  frequencies
    * sample  $\tilde{\omega}_i^1 \sim q_1(\omega|\mathbf{y})$  %  $q_1$  is the proposal distribution, based on FT of  $\mathbf{y}$ 

% Iterations of the MH algorithm

– For  $l = 2$  to  $l_{\max}$ ,
    * For  $i = 1$  to  $I$ , % Note: this is a one-at-a-time algorithm
        . Set  $\omega^* \leftarrow \tilde{\omega}^{l-1}$ 
        . Sample  $\omega_i^* \sim q_2(\omega_i|\tilde{\omega}_i^{l-1}) = \mathcal{N}(\omega; \tilde{\omega}_i^{l-1}, \sigma_{\text{RW}}^2)$ 
        . % Computation of the MH ratio
        . Compute  $p(\omega^*|\mathbf{y})$ 
        . Compute  $p(\tilde{\omega}^{l-1}|\mathbf{y})$ 
        . Finally, compute  $\alpha$  as given in Eq. (19), noting that  $q_2(\omega_i|\tilde{\omega}_i^{l-1}) = q_2(\tilde{\omega}^{l-1}|\omega)$ 
        . % Accept/reject test
        . Sample  $u \sim \mathcal{U}[0; 1]$  %  $u$  is a uniform random variable on  $[0; 1]$ 
        . If  $\alpha > u$ , Set  $\tilde{\omega}^l \leftarrow \omega^*$  % the candidate is accepted
        . Otherwise, Set  $\tilde{\omega}^l \leftarrow \tilde{\omega}^{l-1}$  % the candidate is accepted
```

B Matlab functions provided

Some useful built-in Matlab functions.

Function name	What it does	Observations
<code>rand</code>	Samples a (vector of independent) uniform random variable(s) on $[0; 1]$	
<code>randn</code>	Samples a (vector of independent) zero-mean Gaussian random variable(s) with variance 1	
<code>hist</code>	Plots the histogram of a vector of data	

Custom Matlab functions.

Function name	What it does	Observations
<code>mat_D</code>	Builds the matrix $\mathbf{D}(I, \omega)$	Defined in Eq. (3)
<code>mat_S</code>	Computes the matrix \mathbf{S}	Defined in Eq. (10)
<code>mat_P</code>	Computes the matrix \mathbf{P}	Defined in Eq. (14)
<code>vect_mu</code>	Computes the vector $\boldsymbol{\mu}$	Defined in Eq. (11)
<code>q_1</code>	Generates a (vector of) sample(s) drawn according to the proposal q_1	
<code>val_q1</code>	Computes the value of the pdf q_1 corresponding to a given sample	
<code>q_2</code>	Generates a (vector of) sample(s) drawn according to the proposal q_2	
<code>p_omega_y</code>	Computes the value of the posterior for a given $\boldsymbol{\omega}$	Defined in Eq. (12)
<code>sample_beta</code>	Samples the amplitudes according to their conditional density	Defined in Eq. (15)
<code>sample_sigma</code>	Samples the noise variance according to their conditional density	Defined in Eq. (16)
<code>generesig</code>	Generates a time series composed of sinusoids in noise	See Eq. (2)
<code>start</code>	Defines simulation parameters, launches the MCMC sampling and displays results	You need to change the name of the sampling algorithm called in it
<code>algo_template</code>	Template algorithm (to be filled in!) to sample the Markov Chain one at a time	