

# Unsupervised Learning

## Lecture 7: Markov Chain Monte Carlo

**Zoubin Ghahramani**

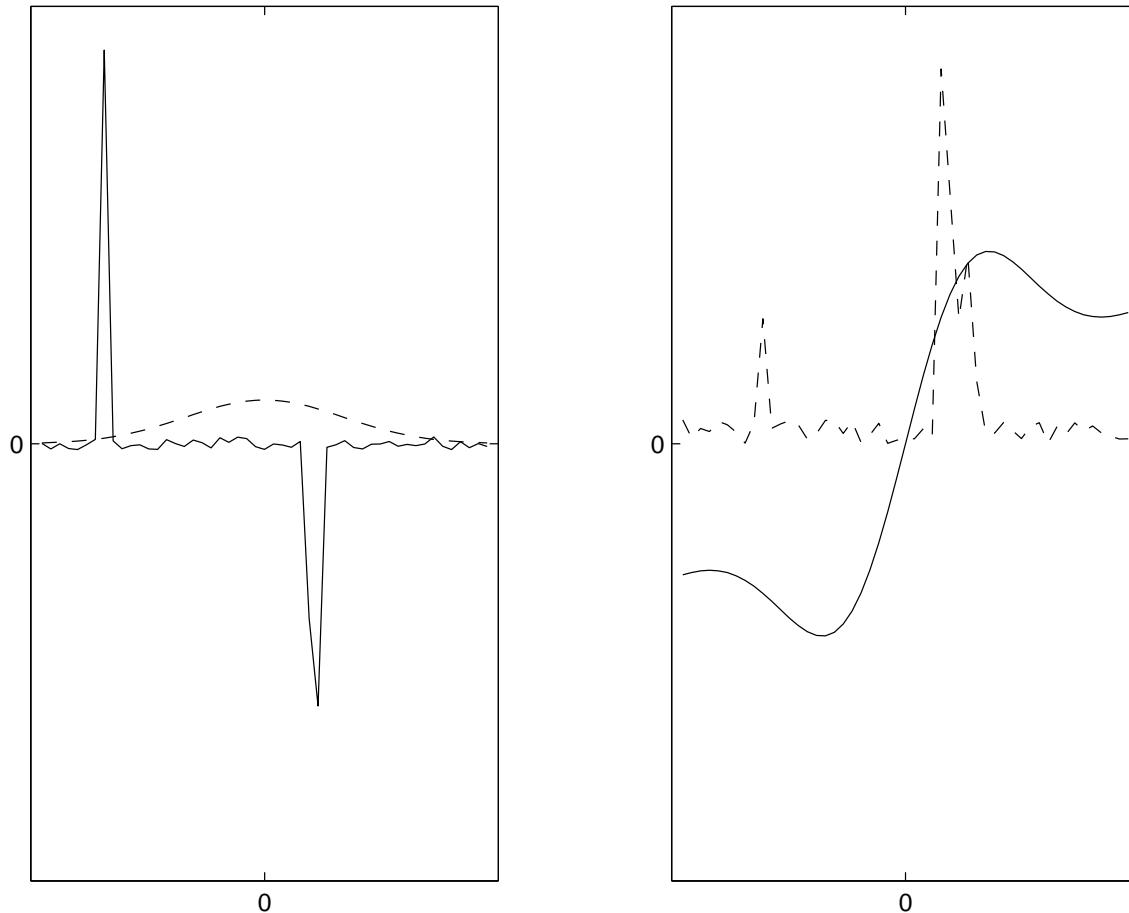
`zoubin@gatsby.ucl.ac.uk`

**Carl Edward Rasmussen**

`edward@gatsby.ucl.ac.uk`

**Gatsby Computational Neuroscience Unit  
MSc in Intelligent Systems, Fall 2001**

# The integration problem



We make computations of the form

$$\int F(w)p(w)dw,$$

where  $F(w)$  is some (probably complicated) function of the model parameters and  $p(w)$  is a probability density.

Two typical situations; left panel: full line is some function times the likelihood, dashed is prior; right panel: full line is some function and dashed is posterior.

## Simple approaches to the problem

Analytical approximation: expansion around the mode (ie Gaussian).

Numerical methods?

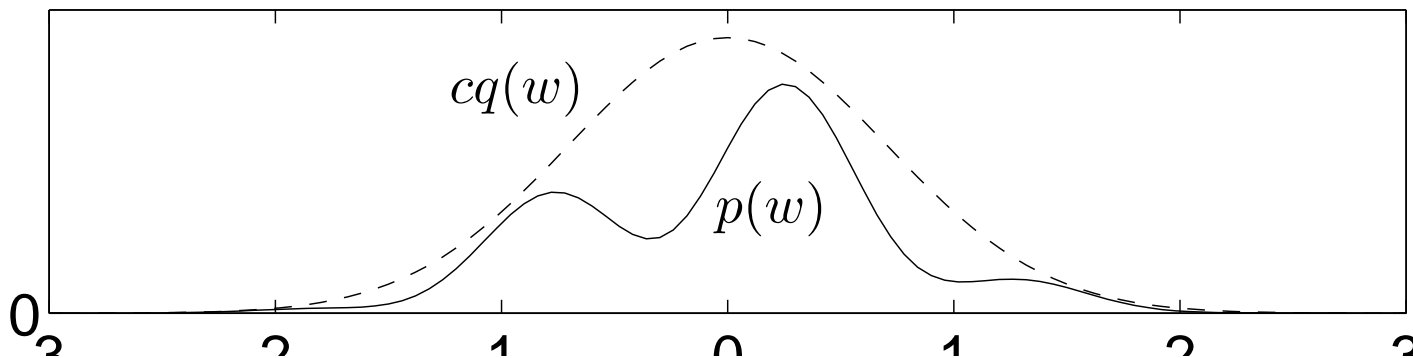
Simple Monte Carlo:

$$\int F(w)p(w)dw \simeq \frac{1}{T} \sum_{t=1}^T F(w^{(t)}),$$

where  $w^{(t)}$  are (independent) samples from  $p(w)$ .

Problem: it may be difficult to obtain the samples from  $p(w)$ .

Simple rejection sampling: Find a distribution  $q(w)$  and a constant  $c$  such that  $\forall w, p(w) \leq cq(w)$ . Sample  $w^*$  from  $q(w)$  and accept  $w^*$  with probability  $p(w^*)/(cq(w^*))$ . Use accepted points in simple Monte Carlo.



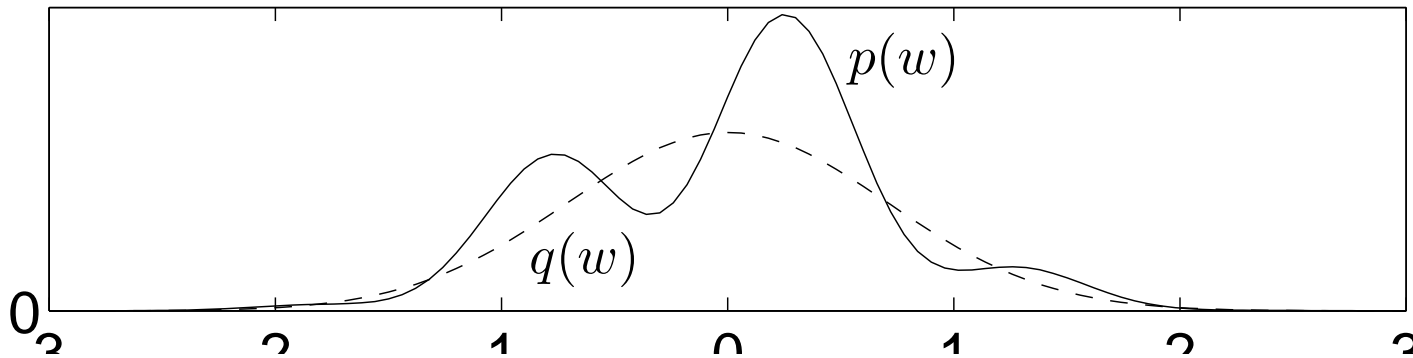
Problem: it may be difficult to find  $cq(w)$  with a small  $c$  which is easy to sample from.

## More simple approaches

Simple importance sampling:

$$\int F(w)p(w)dw \simeq \frac{1}{T} \sum_{t=1}^T F(w^{(t)}) \frac{p(w^{(t)})}{q(w^{(t)})},$$

where  $q(w)$  is non-zero wherever  $p(w)$  is.



Problem: it may be difficult to find a suitable  $q(w)$ . Monte Carlo average may be dominated by few samples (high variance); or none of the high weight samples may be found!

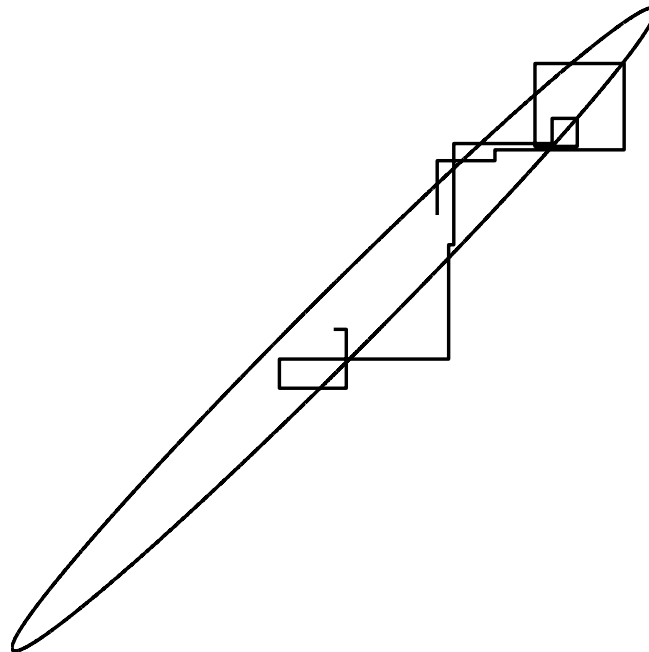
# Gibbs sampling

Give up the requirement that samples should be independent. Repeatedly:

- 1) Pick  $i$  (either at random or in turn)
- 2) replace  $w_i$  by a sample from the conditional distribution

$$p(w_i | w_1, \dots, w_{i-1}, w_{i+1}, \dots, w_n)$$

Gibbs sampling is feasible if it is easy to sample from the conditional probabilities.



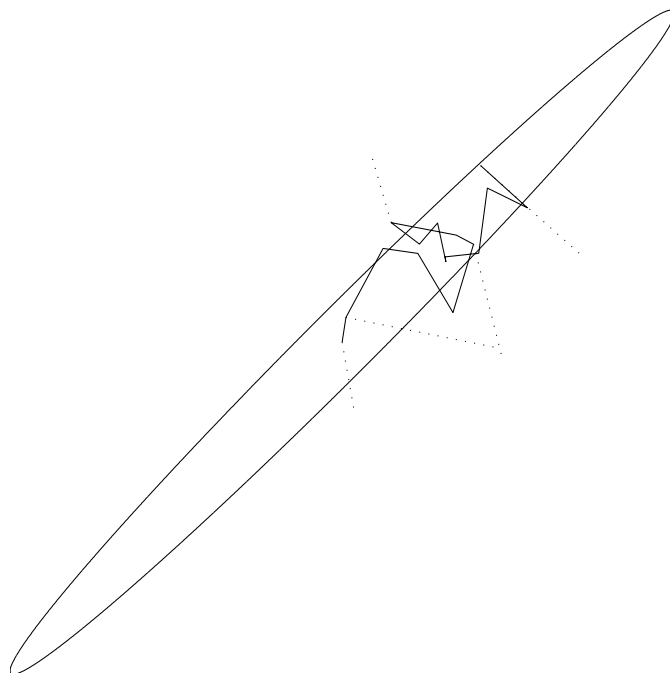
Example: 20 (half-) iterations of Gibbs sampling on a bivariate Gaussian

# The Metropolis algorithm

Based on the current state  $w$ , propose a new state  $w^*$  using a proposal distribution  $S(w, w^*)$ . Accept the new state with probability  $\min(1, p(w^*)/p(w))$ ; otherwise retain the old state.

Proof of correctness relies on symmetry of  $S(w, w^*)$  and detailed balance.

Note, we need only to compute ratios of probabilities (no normalising constants).



Example: 20 iterations of global metropolis sampling from bivariate Gaussian; rejected proposals are dotted

# Hybrid Monte Carlo

Typical distance traveled by a random walk in  $n$  steps is proportional to  $\sqrt{n}$

Fundamental requirement: seek regions of high probability while avoiding random walks

Hybrid Monte Carlo: We introduce a fictitious physical system where model parameters correspond to positions  $q$ ; the position variables are augmented with momentum variables  $p$  to help avoid random walks.

We simulate the dynamical system in such a way that the distributions of positions,  $p(q)$ , corresponds to the desired distribution  $p(w)$ . For estimation purposes we ignore the momentum variables  $p$ .

# Dynamical system

In the physical system, positions  $q$  correspond to weights  $w$  and are augmented by momentum variables  $p$ :

$$p(p, q) \propto \exp(-H(q, p)) \quad H(p, q) = E(q) + K(p)$$
$$E(q) = -\log(p(q)) - \log(Z) \quad K(p) = \frac{1}{2} \sum_i p_i^2 / m_i$$

The physical system evolves at constant energy  $H$  according to Hamiltonian dynamics:

$$\frac{dq_i}{dt} = \frac{\partial H}{\partial p_i} = \frac{p_i}{m_i} \quad \frac{dp_i}{dt} = -\frac{\partial H}{\partial q_i} = -\frac{\partial E}{\partial q_i}.$$

The system is simulated approximately via a series of  $L$  discrete leapfrog steps of size  $\epsilon$ :

$$\hat{p}_i(t + \frac{\epsilon}{2}) = \hat{p}_i(t) - \frac{\epsilon}{2} \frac{\partial E(\hat{q}(t))}{\partial q_i}$$

$$\hat{q}_i(t + \epsilon) = \hat{q}_i(t) + \epsilon \frac{\hat{p}_i(t + \frac{\epsilon}{2})}{m_i}$$

$$\hat{p}_i(t + \epsilon) = \hat{p}_i(t + \frac{\epsilon}{2}) - \frac{\epsilon}{2} \frac{\partial E(\hat{q}(t + \epsilon))}{\partial q_i}$$

# Properties of the dynamical system

Hamiltonian dynamics has the following important properties:

- 1) preserves total energy,  $H$ ,
- 2) is reversible in time
- 3) preserves phase space volumes (Liouville's theorem)

The leapfrog discretization only approximately preserves the total energy  $H$ , and

- 1) is reversible in time
- 2) preserves phase space volume

The dynamical system is simulated using the leapfrog discretization and the new state is used as a proposal in the Metropolis algorithm to eliminate the bias caused by the leapfrog approximation

# Stochastic dynamics

Changes in the total energy,  $H$ , are introduced by interleaving the deterministic leapfrog transitions with stochastic updates of the momentum variables.

Since the distribution of the momenta are independent Gaussian, this is easily done using Gibbs sampling with persistence:

$$p_i = \alpha p_i + \sqrt{1 - \alpha^2} \varepsilon,$$

where  $\varepsilon \sim \mathcal{N}(0, 1)$  and the persistence  $0 \leq \alpha < 1$ .

Persistence is added to further suppress random walks

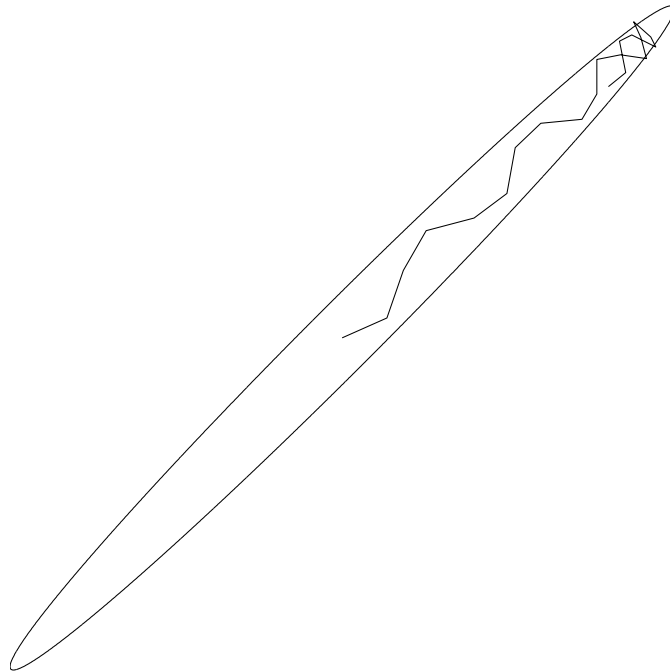
# Hybrid Monte Carlo algorithm

1) A new state is proposed by deterministically simulating a trajectory with  $L$  discrete steps from  $(q, p)$  to  $(q^*, p^*)$ . The new state is accepted by the Metropolis algorithm with probability:

$$\min(1, \exp(-(H(p^*, q^*) - H(p, q))))),$$

otherwise the state remains the same and momentum is negated.

2) Stochastically update the momenta using Gibbs sampling.



Example:  $L = 20$  leapfrog iterations when sampling from bivariate Gaussian

# The neural network model

Training set:  $D = \{(x^{(c)}, t^{(c)})\}$ ,  $c = 1 \dots n$ .

Single hidden layer net (with direct connections from inputs to output):

$$f(x) = \sum_i h_i(x)w_i + \sum_j w_jx_j + b$$

$$h_i(x) = \tanh\left(\sum_j w_{ji}x_j + b_j\right)$$

model parameters ( $w$ 's and  $b$ 's) are called  $\theta$ .

Likelihood (really  $p(y^{(1)}, \dots, y^{(n)} | x^{(1)}, \dots, x^{(n)}, \theta)$ ):

$$p(D|\theta) \propto \prod_{c=1}^n \exp(-(f(x^{(c)}) - t^{(c)})^2 / 2\sigma^2).$$

The common “maximum likelihood” approach minimizes the cost function  $-\log(p(D|\theta))$ .

Task: predict  $y^{(n+1)}$  given a new input  $x^{(n+1)}$  and the training examples  $D$ .

## Using Bayes' rule

Bayes' rule: posterior  $\propto$  likelihood  $\times$  prior.

I wish to use a hierarchical prior, which is specified in terms of “hyperparameters”  $\gamma$ :

$$p(\theta, \gamma | D) \propto p(D | \theta) p(\theta | \gamma) p(\gamma).$$

The predictive distribution is obtained by integration:

$$p(y^{(n+1)} | x^{(n+1)}, D) = \int p(y^{(n+1)} | x^{(n+1)}, \theta, \gamma) p(\theta, \gamma | D) d\theta d\gamma.$$

Mean of predictive distribution (for squared error loss):

$$\hat{y}^{(n+1)} = \int f(x^{(n+1)}, \theta) p(\theta, \gamma | D) d\theta d\gamma.$$

Computational task: do the integrals over the posterior distribution.