ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

HASCO SUMMER SCHOOL 2013

Bayesian Inference in Processing Experimental Data Principles and Basic Applications

arXiv:physics/0304102v1 [physics.data-an] 28 Apr 2003

Author: G. D'Agostini

Presented by:
Federica Fabbri
Marco Fiore

Outline

- Bayesian Inference;
- Computational issues;
- Sampling;
- Markov Chain Monte Carlo (MCMC);
- Conclusions;
- Q & A (or not)

Introduction

The subjective nature of probability ("*…a quantitative measure of the strength of our conjecture or anticipation, founded on the said knowledge, that the event comes true" E. Schroedinger)* is well represented by:

$$P(H_j \mid E_i, I) = \frac{P(E_i \mid H_j, I) P(H_j \mid I)}{\sum_j P(E_i \mid H_j, I) P(H_j \mid I)}$$

Bayes' Theorem

Introduction

The subjective nature of probability ("*…a quantitative measure of the strength of our conjecture or anticipation, founded on the said knowledge, that the event comes true" E. Schroedinger)* is well represented by:

$$P(H_j \mid E_i, I) = \frac{P(E_i \mid H_j, I) P(H_j \mid I)}{\sum_j P(E_i \mid H_j, I) P(H_j \mid I)}$$

Bayes' Theorem

$$P(\theta \mid d, I) = \frac{P(d \mid \theta, I) P(\theta \mid I)}{\int P(d \mid \theta, I) P(\theta \mid I) d\theta}$$

Everything we do is based on what we know about the world.

Computational issues 1

The application of Bayesian ideas leads to computational problems, mostly related to the calculation of integrals for:

Computational issues 1

The application of Bayesian ideas leads to computational problems, mostly related to the calculation of integrals for:

Normalization $\int p(x)\,dx = 1$

Computational issues 1

The application of Bayesian ideas leads to computational problems, mostly related to the calculation of integrals for:

Normalization $$\int p(x)\,dx = 1$$

Expectation value of f(X) $$E[f(X)] = \int f(x)\,p(x)\,dx$$

Computational issues 1

The application of Bayesian ideas leads to computational problems, mostly related to the calculation of integrals for:

Normalization $$\int p(x)\,dx = 1$$

Expectation value of f(X) $$E[f(X)] = \int f(x)\,p(x)\,dx$$

Marginalization $$\int p(x,y)\,dy = p(x)$$

Computational issues 2

If we were able to *sample* the posterior, i.e. to generate points of the parameter space according to their probability, the problem would have been solved, at least approximately.

$$E[f(x)] = \sum_{x_1} \ldots \sum_{x_n} f(x_1, \ldots, x_n) p(x_1, \ldots, x_n) = \sum_i f(\bar{x}_i) p(\bar{x}_i)$$

Computational issues 2

The average $\langle f(x) \rangle$ can then be calculated with

$$\langle f(x) \rangle = \frac{1}{N} \sum_t f(\bar{x}_t)$$

For simple distributions (Gaussian, Poisson, Breit-Wigner) there are well-known standard techniques for generating pseudorandom numbers starting from numbers distributed uniformly between 0 and 1.

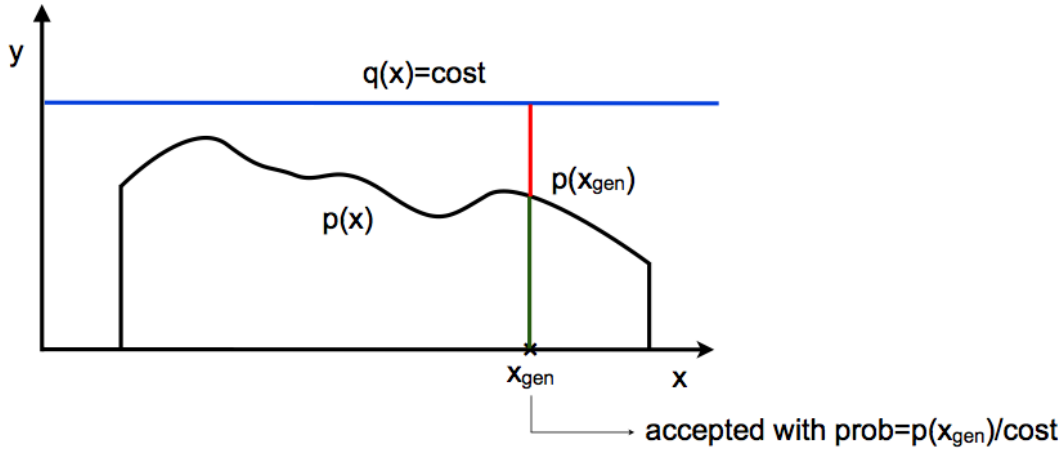
Sampling 1

If the distribution p(**x**) you use to sample is complicated, you need a different approach:

Sampling 1

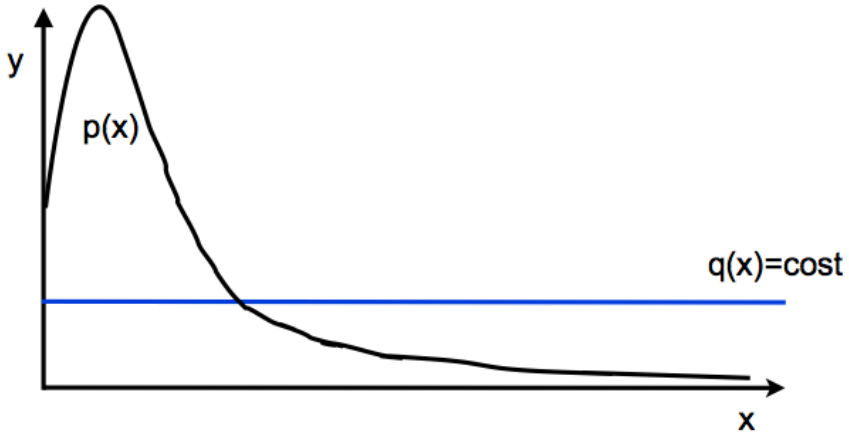
If the distribution p(**x**) you use to sample is complicated, you need a different approach:

Rejection sampling: We generate $\mathbf{x}_{gen}$ according to some function q(**x**), such that, given a constant c, p(**x**)≤cq(**x**) and decide to accept it with probability p($\mathbf{x}_{gen}$)/cq($\mathbf{x}_{gen}$).



Sampling 2

Importance sampling : even without requirements on the g(**x**) function, apart from the condition that g($\mathbf{x}_i$) must be positive wherever p($\mathbf{x}_i$) is positive, we can still use it to calculate E[f(**x**)]:



$$E[f(x)] \approx \frac{\sum_t f(x_t)\,\tilde{p}(x_t)/g(x_t)}{\sum_t \tilde{p}(x_t)/g(x_t)}$$

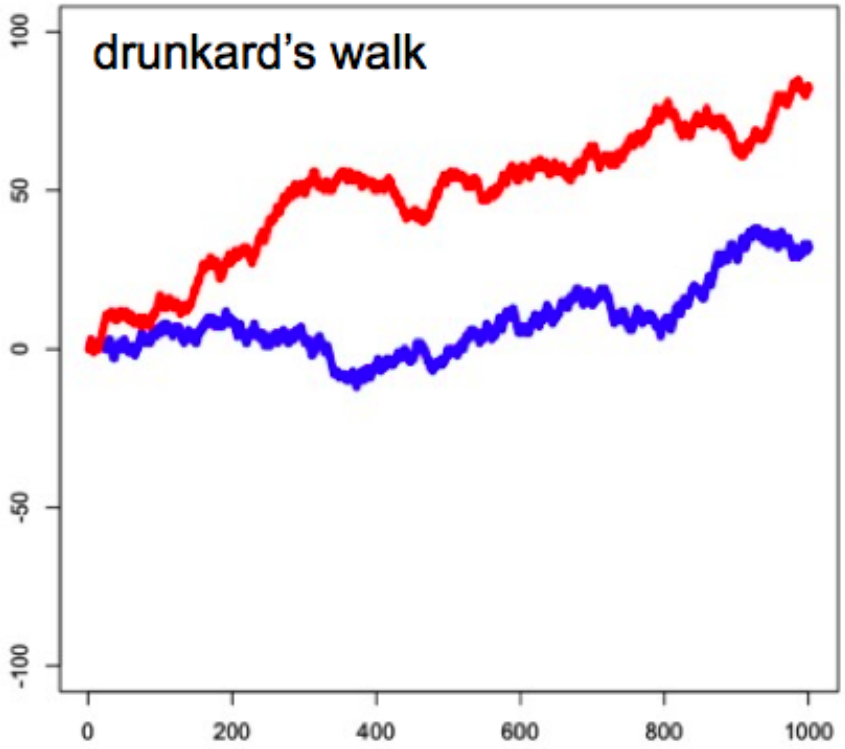
In this case the sampling produces weighted events.

Sampling: MCMC

A different class of Monte Carlo methods is based on Markov Chains: the sequence of generated points takes a kind of random walk in parameter space, instead of each point being generated one independently from another.

Sampling: MCMC

A different class of Monte Carlo methods is based on Markov Chains: the sequence of generated points takes a kind of random walk in parameter space, instead of each point being generated one independently from another.



The probability of jumping from one point to an other depends only on the last point and not on the entire previous history.

Metropolis algorithm

One of the most popular and simple algorithms, based on MCMC;

Start from an arbitrary point x_0 and generate the sequence by repeating the following cycle:

1. Select a new trial point **x*** chosen according to a symmetric *proposal* pdf $q(\mathbf{x}^*|\mathbf{x}_t)$
2. Calculate the *acceptance probability*

$$A(x^* \mid x_t) = \min[1, \frac{\tilde{p}(x^*)}{\tilde{p}(x_t)}]$$

Metropolis algorithm

One of the most popular and simple algorithms, based on MCMC;

Start from an arbitrary point x_0 and generate the sequence by repeating the following cycle:

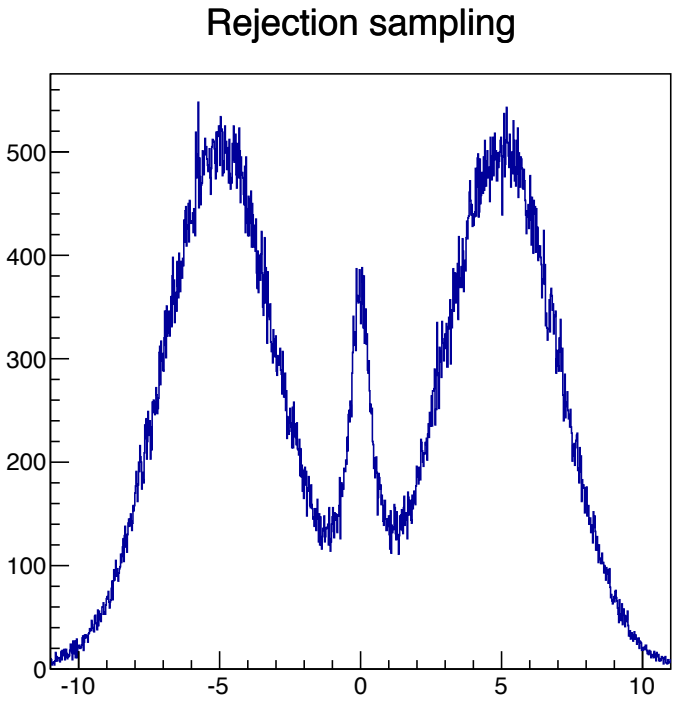
1. Select a new trial point $\mathbf{x}^*$ chosen according to a symmetric *proposal* pdf $q(\mathbf{x}^*|\mathbf{x}_t)$
2. Calculate the *acceptance probability;*
3. Accept $\mathbf{x}^*$ with probability $A(\mathbf{x}_t,\mathbf{x}^*)$, i.e.
   - If $\tilde{p}(x^*) \geq \tilde{p}(x_t)$, then accept x*;
   - If $\tilde{p}(x^*) < \tilde{p}(x_t)$, accept x* if a number, extracted randomly between 0 and 1, is less than $\tilde{p}(x^*) / \tilde{p}(x_t)$

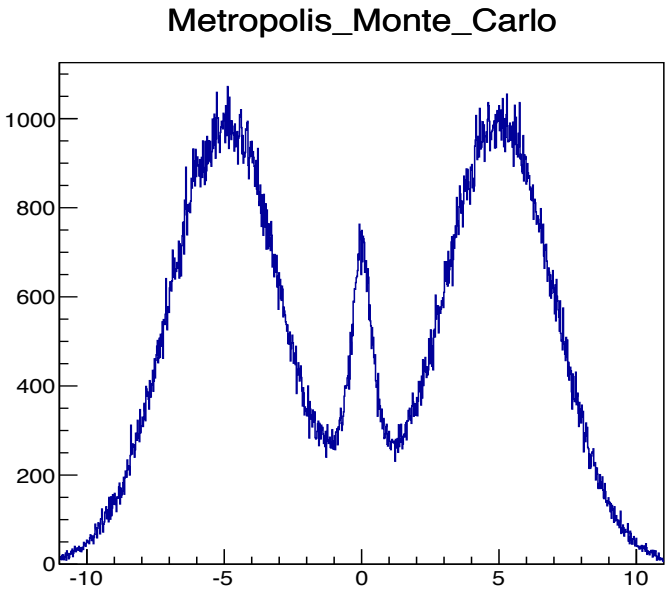
   If the point is accepted, then $\mathbf{x}_{t+1} = \mathbf{x}^*$. Otherwise, $\mathbf{x}_{t+1} = \mathbf{x}_t$

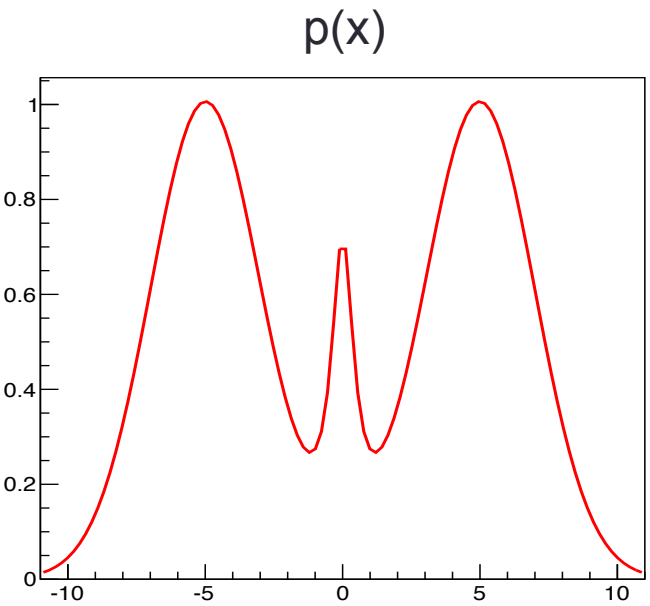
Examples

$\varepsilon_{g(\mathbf{x})}$[Metropolis] = 0.416264

$\varepsilon_{g(\mathbf{x})}$[rejection] = 0.249386





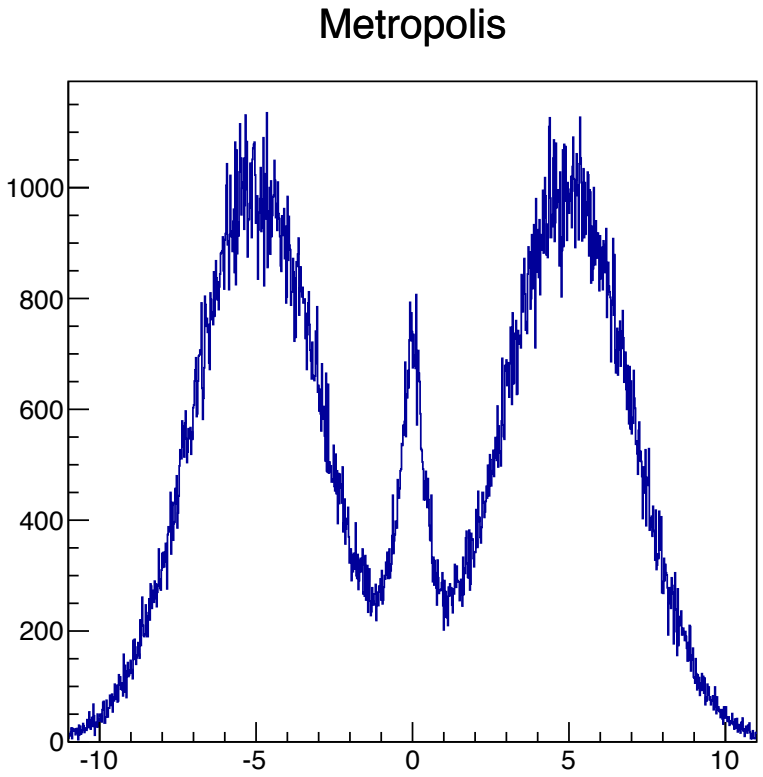


Problems

- Each point in the chain has some correlation with the points which immediately preceded it, and usually the chain moves slowly from one region in the variable space to another and it can happen several times that $\mathbf{x}_{t+1} = \mathbf{x}_t$;
- The initial part of the sequence is strongly influenced by the arbitrary starting point. Therefore, it is necessary to remove the initial part of the chain.
- One of the important things to choose with care is the proposal function. If too small jumps are proposed, the chain can even remain trapped in a subregion; if the jumps are too large, the chain remains stuck in a point for many cycles.

Problems

$\varepsilon_{g(\mathbf{x})}$[Metropolis] = 0.191088



One of the important things to choose with care is the proposal function. If too small jumps are proposed, the chain can even remain trapped in a subregion; if the jumps are too large, the chain remains stuck in a point for many cycles.

Conclusions

- The Bayesian approach can be applied to many physical problems (even n-dimensional);
- Providing an exact solution for inferential problems can easily lead to computational difficulties;
- Monte Carlo methods allow to overcome issues in the analysis;
- Metropolis algorithm, based on Markov Chains, gives more accurate (and efficient) results then, for instance, the rejection sampling. Also easy to implement.
- …the end?

Simulated Annealing

- For certain problems, simulated annealing (heating and controlled cooling of a material to increase the size of its crystals and reduce their defects) may be more efficient than exhaustive enumeration, provided that the goal is merely to find an acceptably good solution in a fixed amount of time, rather than the best possible solution.
- Like a decrease in the thermodynamic free energy, in the SA algorithm there is a decrease in the probability of accepting worse solutions→more extensive search for the optimal solution.
- Gradual reduction of the pseudotemperature T as the simulation proceeds: the algorithm starts initially with T set to a high value (or infinity), and then it is decreased at each step following some *annealing schedule* - which may be specified by the user, but must end with T = 0.

LA TOMBA DI GAUSS