Monte Carlo Methods, Courant Institute, Spring 2017

http://www.math.nyu.edu/faculty/goodman/teaching/MonteCarlo17/

Always check the class message board on the NYU Classes site from home.nyu.edu before doing any work on the assignment.

Assignment 2. Due march

Corrections: none yet.

This assignment illustrates several parts of Monte Carlo practice. The overall problem is a Bayesian fitting problem, to find the parameters $A_1, \ldots, A_m$ and $\lambda_1, \ldots, \lambda_m$ to produce a model

$$f(t) = \sum_{i=1}^{m} A_i e^{-\lambda_i t} \ .$$

The $2m$ parameters are to be *trained* in a Bayesian sense from observations

$$f(t_i) = Y_i + \sigma \xi_i \ , \quad i = 1, \ldots, N \ .$$

We assume that the ξ are i.i.d. $\mathcal{N}(0,1)$, so the observation errors have a common variance σ^2, which also is unknown. Overall, there are $n = 2m + 1$ unknowns. These consist of the $2m$ variables we are interested in and the *nuisance* variable σ. This is a variable we are not interested in but which we have to sample in the MCMC.

The assignment has several stages, each of which you need to code. Each task should be in a separate Python module. The main module should `import` the others.

1. Choose true parameters and create fake data. To create fake data, you choose times t_j, evaluate $f(t_j)$ exactly, add Gaussian noise, and record the "noisy observations". The input data to the MCMC code is the vectors with components t_j and Y_j.

2. Given the data and a set of parameters, evaluate the likelihood function (formulas from assignment 1).

3. A procedure to take one MCMC step. That means, propose some changes in the parameters and do the Metropolis Hastings accept/reject. The easiest MCMC algorithm to program is isotropic Gaussian proposal with length scale r in each variable. You will see that this doesn't work very well for hard problems.

4. Do a long run with many MCMC steps to create a collection of samples of the posterior.

5. Evaluate auto-correlation functions to see the auto-correlation times of the parameters.

6. Visualize the results by making histograms and/or scatterplots. There is a scatterplot facility in matplotlib.

The assignment will be vague, leaving many decisions to the individual doing it. You will be graded on the choices you make. If you do the least possible thing, that will get a lower score. If you write Python scripts to produce enormous amounts of data without much thought, that also will get a low score. To get a higher score, explore some issue and try to find something interesting. This could be about correlations in the posterior distribution, or in the autocorrelation functions, or in good proposal distributions, or in how things scale with m, or on how things depend on the choice of the observation points t_i or on the true parameters. It's up to you.

In order to get a good grade, you must make your software as automated as possible. You must be able to do repeated numerical experiments and visualizations with a minimal amount of typing. Otherwise, the explorations required to get a good grade will be very very time consuming, repetitive, and frustrating.