Familiarise yourself with the HMC sampling algorithm

Code to implement HMC is supplied in the module hmc_Lab.py and there is a simple demonstration of its usage in the notebook demo_hmc.ipynb.

Included in hmc_Lab.py are the functions:

- sample: The main function, to return samples from some arbitrary distribution over a state space (e.g. a posterior over combined model parameters **θ**)
- one_step: Implements a single cycle of HMC (you needn't call this directly)
- gradient_check: A useful helper to check your gradient calculations (can also be called automatically from within sample). Highly recommended!

Before undertaking any subsequent tasks, you should examine and run the example notebook demo_hmc.ipynb, which offers a simple guide on to how to set up and use the sample algorithm.

The sampler is called as:

sample(x, energy_func, energy_grad, R, L, eps, burn=0, checkgrad=False, args=())

Arguments are:

- x: The initial state (parameter values) of **θ**. In this exercise, any sensible random initialisation should be fine
- energy_func: The energy (negative log-probability) of the distribution to be sampled from
- energy_grad: Gradient of the above energy function with respect to **θ**
- R: Number of samples desired (try a few hundred for testing, then perhaps up to 10,000 for quoting results)
- L: Number of steps to run the Hamiltonian dynamics within each cycle of the sampler (L=100 is probably safe, though for the Gaussian example below, 25 should suffice)
- eps: Step-size for the dynamics (see below)
- burn: Number of initial samples to discard as 'burn-in' (e.g. you might set this to 10% of R)
- checkgrad: A flag specifying whether to check the consistency of the error/gradient functions – it is recommended to set this to True!
- args: A list containing any additional (fixed) arguments that need to be passed to both energy_func and energy_grad (later, you will need to pass the data inputs and targets)

Key to the accuracy of the sampler are the 'simulation' parameters L and eps. The former should be set large enough such that the dynamics have time to travel the full range of the distribution for a given step-size eps, which itself should be sufficiently small to ensure the simulation does not 'diverge'. The best settings are rarely obvious without some experimentation!

Here, the general guidance is:

- Set L to 100 (as a minimum) for most regression exercises
- As an exception, setting L to 25 for the simple Gaussian example is sufficient
- Feel free to experiment and use a smaller L when testing
- Given choice of L, eps should be set to be as large as possible while keeping the acceptance rate of the sampler relatively high, e.g. no lower than 80%
- Note that having an acceptance rate of 100% is not likely to be a good thing. That may indicate that eps is too small and the state space is not being properly explored
- One informal approach may therefore be to increase eps until the acceptance rate falls below 80%, then ‘dial it back down a little’.