Lecture 12, Friday w6, 2014-10-24

Markov chain Monte Carlo (MCMC)

Distributions that appear in physics and machine learning are often of the form $p(x) = p^*(x)/Z$, where we can evaluate $p^*(x)$ at any setting x, but we can't feasibly compute the normalizing constant Z.

Example 1: The posterior over parameters of a complicated model $p(\theta|D) \propto p(D|\theta)p(\theta)$. There may be many parameters, and the posterior distribution is often not in a standard form that we can sample from directly.

Example 2: An Ising model (you do not need to know the form of this model for this course). A model of binary images, where each pixel $x_i \in \{-1, +1\}$ is influenced by its neighbours. Coupling strengths W_{ij} give the strength of interactions between neighbours. $P(x|W) = \exp(-\sum_{ij} W_{ij}x_ix_j)/Z(w)$. For large such models it isn't at all obvious how to sample from these distributions. However, we can compare the ratio of probabilities of two images x and x'.

Markov chain Monte Carlo constructs a process that explores the target distribution. In the examples: 1) plausible parameters given data; 2) typical images under the model. At each time step the chain is at one particular state, "maybe your data was generated from these parameters", or "this is a reasonable image". At the next time step it will move to a slightly different reasonable setting, similar to the last (in Metropolis methods, sometimes exactly equal to the last).

Stationary condition: given a sample from the target distribution, the algorithm draws another point that is also marginally from the target distribution. Although it is dependent on the previous point: the parameters or image won't change very much from one iteration to the next. Know and be able to explain the equation. (I think if you understand it, you should remember it.)

Ergodicity: We won't usually be able to even draw one distribution from our target distribution as a starting point. However, if we sample the initialization from some arbitrary distribution, the distribution over where the chain ends up after t steps will tend towards the correct distribution. There are technical conditions: it must be possible for the chain to get anywhere in the state space in finite time. (The details and a proof are beyond this course. There's more at the end for those interested.)

Methods

Gibbs sampling: Sampling from high-dimensional probability distributions is hard, so sample from one-dimensional distributions instead. The conditional may be in a standard form (discrete/categorical, Gaussian, gamma, …). If not, we may be able to use rejection sampling.

Metropolis–Hastings sampling: Use an arbitrary proposal that can depend on the current state. Usually a small perturbation of the current state. Accept move $x \to x'$ with probability $\min(1, p(x')q(x; x')/((p(x)q(x'; x))))$. For symmetric proposals, $q(x; x') = q(x'; x)$, like a Gaussian centered on current state, get simpler Metropolis acceptance probability: $\min(1, p(x')/p(x))$. If reject proposal, copy previous state: set $x_{t+1} = x_t$.

Setting step sizes can be important. Too small, slow progress. If half the step size can make progress four times slower! If proposals too broad, the chain hardly ever moves. (An acceptance rate of around 0.234 is considered optimal, but does not need to be precisely tuned. The method is valid for any step size.)

Recommended reading

Again there are three alternatives. Any of which is fine, depending on which style you prefer.

A terse summary of this material continues in my PhD thesis, pp24–30. http://homepages.inf.ed.ac.uk/imurray2/pub/07thesis/

MacKay's textbook: continue from last time's reading to pp365–374.

Murphy's textbook: Chapter 24 pp837–839, pp848–852.

Non-examinable extras

I've been asked a lot of questions by keen students. So there's a lot of pointers here. But far more than most of you will have time to look at. You don't need to look at any of the stuff below! Focus on one of the readings above and the Tutorial exercises and you should be fine.

I was asked after the lecture for examples. There is a simple one in the slides. There is another one in the tutorial this week (which will come with an answer). The assessed assignment has another, which is in the optional part so you can ask me about it now. A full answer will be provided. As an illustrative example, you could also try applying the code to Bayesian linear regression with Gaussian noise in (even though it doesn't need MCMC). It should only be a few lines of code. I'll happily help if you get stuck. Yet another demo is in an exercise here: http://homepages.inf.ed.ac.uk/imurray2/teaching/09mlss/

So you know, there is classic software to automatically Gibbs sample from models you can specify with a flexible modelling syntax (WinBUGS, JAGS, …). The modern term is *probabilistic programming*: you write code that describes a model, and a clever compiler tries to do inference for you (using MCMC, or otherwise).

Some Monte Carlo methods are easy to implement yourself. We saw that a Metropolis implementation fits on a slide. Even fancier methods like slice sampling (next time) and Hamiltonian Monte Carlo (not covering this year) have really short code, the MacKay book contains pseudo-code or Octave code for both. Tuning and diagnosing these methods can be harder. R-CODA is an R package that can help a little with diagnostics.

Advanced reading for future reference:

- http://www.cs.toronto.edu/∼radford/review.abstract.html is an old but excellent review by Radford Neal.
- The sample chapters of a recent (expensive!) book are excellent: http://www.mcmchandbook.net/HandbookSampleChapters.html

You will find more technical details on *ergodicity*, and why MCMC works even if you have an arbitrary starting point, in those references. (Or following the Tierney reference from the reading from my thesis.)

Here's the high-level idea, or one way of looking at it. Let the marginal probability distribution of the state of a Markov chain at time t be $c_t(x)$. If you sample the initial condition from your target distribution, $c_0(x) = p(x)$, then by induction, the stationary condition of a valid MCMC algorithm means that $c_t(x) = p(x)$ for all time. If you instead initialize your algorithm in some arbitrary way, $c_0(x) = q(x)$, then the sequence of distributions $c_1, c_2, ...$ will be different. However, the target distribution is a fixed point in distribution space. If the distribution after many iterations is the target distribution, the distribution will be the target distribution for all future time. The target distribution is also an attractor. Valid MCMC methods will pull you closer towards the target distribution at every step.

For a proof of that last point, see Cover and Thomas's Information Theory textbook, Section 2.9, or bash through my brute force version here: http://homepages.inf.ed.ac.uk/imurray2/pub/08mckl/

Original Metropolis et al. paper: http://dx.doi.org/10.1063/1.1699114

Marshall Rosenbluth's account of who did what: http://dx.doi.org/10.1063/1.1887186