Using Deterministic Maps when Sampling from Complex Distributions

Radford M. Neal

University of Toronto, Vector Institute Affiliate

http://www.cs.utoronto.ca/∼radford

http://radfordneal.wordpress.com

Evolution of Deep Learning Symposium, Toronto, 16 October 2019

Two Foundational Papers From the 1950s

Metropolis, Rosenbluth, Rosenbluth, Teller and Teller, "Equation of State Calculations by Fast Computing Machines".

> Introduced the "Metropolis algorithm" (the first "Markov Chain Monte Carlo" method) — a general way of sampling from complex, high-dimensional distributions. Applied it to simulating a system of molecules. Has been used extensively in physics ever since.
>
> Its applicability to statistical learning went unrecognized for decades.

Alder and Wainwright, "Studies in Molecular Dynamics. I. General Method".

> Simulated molecular systems deterministically. It also has been used extensively in physics ever since, with applications overlapping those of the Metropolis algorithm.

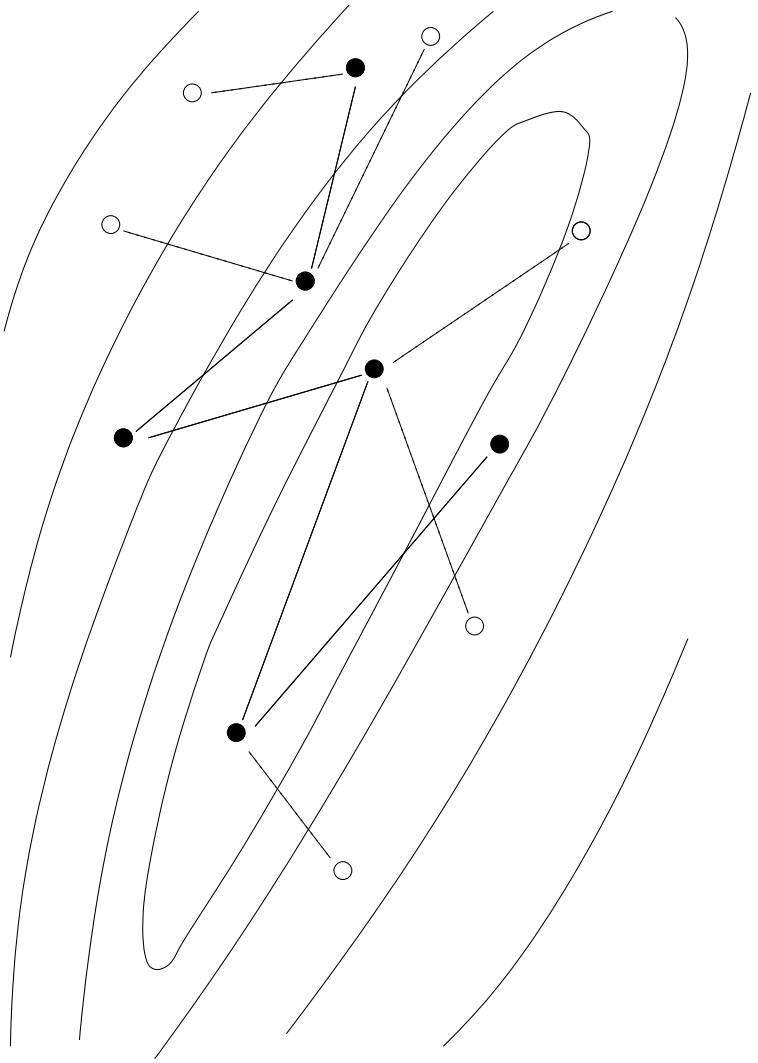
The Metropolis Algorithm

Sample a sequence $x_1, x_2, \ldots$ whose distribution converges to $\pi(x)$.

Use a "proposal" distribution, $q(x^*|x)$, with $q(x|x^*) = q(x^*|x)$.

To update from x_t to x_{t+1}, first propose x^* from $q(\cdot|x_t)$.

With probability $\min[1, \pi(x^*)/\pi(x)]$, set x_{t+1} to x^* ("accept"); otherwise let x_{t+1} remain at x_t ("reject").



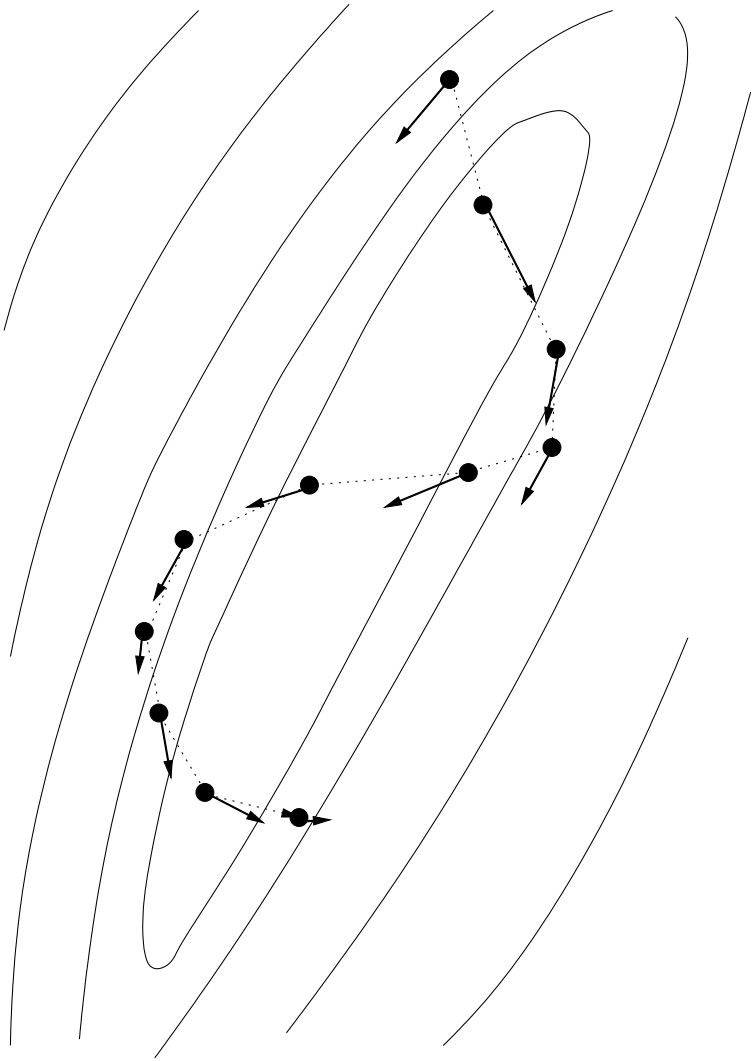
Molecular Dynamics

Simulate the dynamics of molecules moving as influenced by forces between them, and from outside.

For one molecule, with mass 1, position q, and momentum p, we can (approximately) simulate movement from time t to $t + \eta$ with the "leapfrog" method:

$$\begin{aligned} p_{t+\eta/2} &= p_t - (\eta/2)\,\nabla U(x_t) \\ x_{t+\eta} &= x_t + \eta\, p_{t+\eta/2} \\ p_{t+\eta} &= p_{t+\eta/2} - (\eta/2)\,\nabla U(x_{t+\eta}) \end{aligned}$$

If it were exact, this would conserve the total energy (the "Hamiltonian"), $U(x)+K(p)$, with $K(p) = |p|^2/2$ in this example. The approximate simulation *exactly* preserves volume.



Two Foundational Papers From 1986 (in "Parallel Distributed Processing")

Hinton and Sejnowski, "Learning and Relearning in Boltzmann Machines".

> An elegant approach to probabilistic modeling, implemented using Markov chain Monte Carlo methods, before these were commonly known outside physics.
>
> Learning was done by gradient descent, with the gradient evaluated by Monte Carlo, as a difference of "positive" and "negative" phase statistics. This turned out not to work well in practice...

Rumelhart, Hinton, and Williams, "Learning Internal Representations by Error Propagation".

> The classic "backpropagation" paper — the start of a demonstration that gradient descent optimization can do much more than you might have thought.
>
> A dynamical approach of "gradient descent with momentum" was often used.

A Crucial Paper from 1987

Duane, Kennedy, Pendleton, and Roweth, "Hybrid Monte Carlo".

> Merged molecular dynamics and the Metropolis algorithm, using gradient information to greatly improve sampling — dynamical simulation can propose distant points, avoiding slow exploration by a random walk.
>
> Used by physicists doing lattice field theory, a numerical approach to computations for quantum chromodynamics.

The method is now dubbed "Hamiltonian Monte Carlo", since the formulation of dynamics using a "Hamiltonian" function is used.

This is made possible by imagining that x is the position of a "molecule" that has momentum p. We let x and p be independent, with densities proportional to $\pi(x) = \exp(-U(x))$ and $\exp(-K(p))$. After sampling for (x, p), we can just ignore p to get a sample for x.

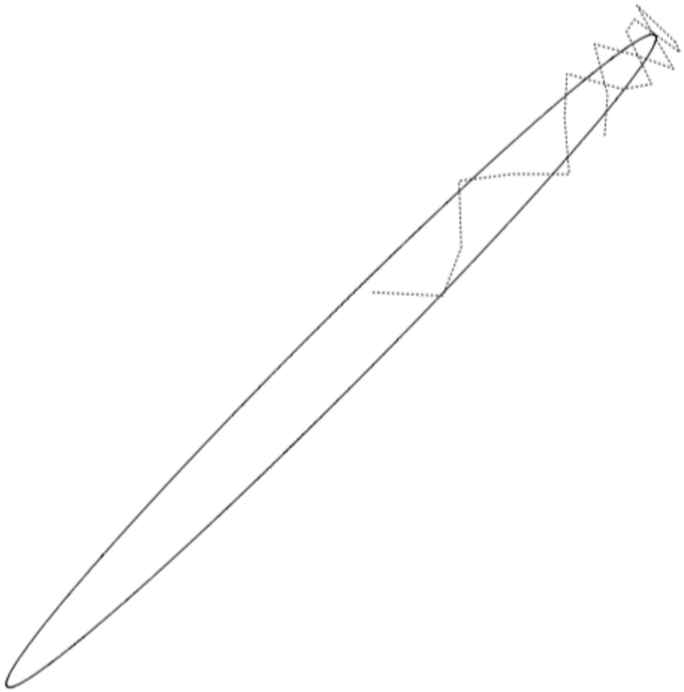
Hamiltonian Monte Carlo

Sample (x_1, p_1), (x_2, p_2), ... converging to a distribution with density proportional to $\exp(-U(x)) \exp(-K(p))$.

Propose a new (x^*, p^*) by simulating dynamics starting at (x_t, p_t) for L leapfrog steps, with stepsize η, then negating p (so the proposal will be reversible).

Accept or reject (x^*, p^*) in Metropolis fashion.

Then sample p_{t+1} according to $\exp(-K(p))$.



An HMC trajectory for a bivariate Gaussian.

The World in 1989

I began my PhD at Toronto with Geoff in Fall 1989.

Geoff and his students were using backprop, and trying to get Boltzmann machines to work better using a "mean field" approach.

Neural networks and similar "connectionist" ideas were a minority approach in AI, with symbolic approaches being dominant. (These two schools of thought seemed to get along fine at U of T, however.)

There was debate about whether probability was of any use in AI, though Hidden Markov Models trained by the EM algorithm were the dominant approach to speech recognition.

There was lots of hype about "expert systems", built with manually codified knowledge, sometimes in terms of manually specified probabilities.

Few ML researchers knew much statistics; fewer statisticians knew of ML.

Bayesian Learning

I was dis-satisfied with the ad hoc nature of machine learning methods, in particular how "overfitting" was avoided only by limiting model complexity.

I found "Bayesian" methods more attractive.

- They express prior beliefs about what might be (eg, how the class of an objects relates to its features) in terms of probabilities, which are updated when data is observed — a philosophically coherent approach.
- I found that they avoid overfitting with large models — even neural network models with *infinitely* many hidden units and parameters give good results, when fitted properly by Bayesian methods.
- Moreover, one can learn "hyperparameters" — high-level parameters that control the distribution of lower-level parameters — and hence learn things like which features are actually relevant to classification.
- Bayesian methods give predictions that quantify uncertainty, since they are based on a distribution over model parameters, not a single estimate.

The Difficulty of Bayesian Learning for Neural Networks

Even modest-size neural networks are much more complex than many "traditional" statistical models. Eg: 10 inputs, 20 hidden units, one output $\rightarrow$ 241 parameters. Of course, "deep" networks can have far more parameters.

Neural network posterior distributions are also hard to visualize, and possibly quite strange. Eg: The maximum likelihood estimate will be non-unique, and have some parameters at infinity (though infinities will be avoided with a proper prior).

The traditional computational methods of Bayesian statistics in 1990 were hopelessly inadequate for such a problem, without resorting to simplifying approximations. Markov chain Monte Carlo methods were only just being looked at by statisticians (eg, Gelfand and Smith's Gibbs sampling paper). The problem is too difficult for simple MCMC methods in any case.

How to do Bayesian learning?

Around when I became interested in Bayesian neural networks, David MacKay did as well. He developed computational methods based on approximating the posterior parameter distribution by a Gaussian, with covariance matrix found from 2nd derivatives at the mode.

Approximations can also be found by minimizing a "variational" criterion.

I preferred to use Markov chain Monte Carlo methods, which are exact asymptotically (but in practice?).

The complexities of neural network posterior distributions perhaps resemble those of distributions found in statistical physics. So it is maybe not surprising that the Hamiltonian Monte Carlo method from physics is useful.

Hamiltonian Monte Carlo Makes Bayesian Neural Networks Practical

A breakthrough for me came when I read about "hybrid" (Hamiltonian) Monte Carlo, and recognized that it was a powerful method applicable beyond its lattice field theory origins — in particular, to Bayesian neural networks.

Three characteristics of HMC make it highly suited to this task:

1. It uses the information in the gradient — which is readily available via backpropagation. In fact, simulation of a trajectory to find a proposal is very similar to standard "backprop with momentum".

2. It can propose distant points by simulating long trajectories, with a high acceptance probability (since the exact dynamics preserves H). This avoids inefficient exploration via a random walk with small steps.

3. It has favourable scaling behaviour with dimensionality, D (with certain assumptions on how dimensionality increases). If computing the density is proportional to D, the time to sample with HMC scales as $D^{5/4}$, much better than the D^2 scaling of simple Metropolis updates.

... But Not Practical Enough for Everyone

Bayesian neural networks implemented with Hamiltonian Monte Carlo perform very well for classification and regression problems of small to moderate size. For example, I used them in the winning entry for the NIPS*2003 feature selection challenge (Neal and Zhang 2006).

But these methods can't be used directly for problems with millions of training cases, or when models with millions of parameters are needed.

However, the idea that with a good learning method, model complexity does not need to be limited to prevent overfitting does extend to such large-scale problems. In particular, "dropout" (Hinton, et al 2012) can be interpreted in Bayesian terms (Gal and Ghahramani 2016).

Other Uses of Deterministic Maps for Sampling?

Hamiltonian Monte Carlo uses a complex deterministic process utilizing gradients to propose a new state, combined with stochastic sampling of momentum and stochastic accept/reject decisions. It can be enormously better than purely stochastic random-walk Metropolis methods.

Are there other ways of usefully incorporating deterministic maps into Markov chain Monte Carlo?

Some possible stochastic aspects that might be made deterministic:

- Metropolis accept/reject decisions.
- Choosing Metropolis proposals.
- Sampling from univariate conditional distributions ("Gibbs sampling").
- Everything.

Taking the Monte Carlo out of MCMC

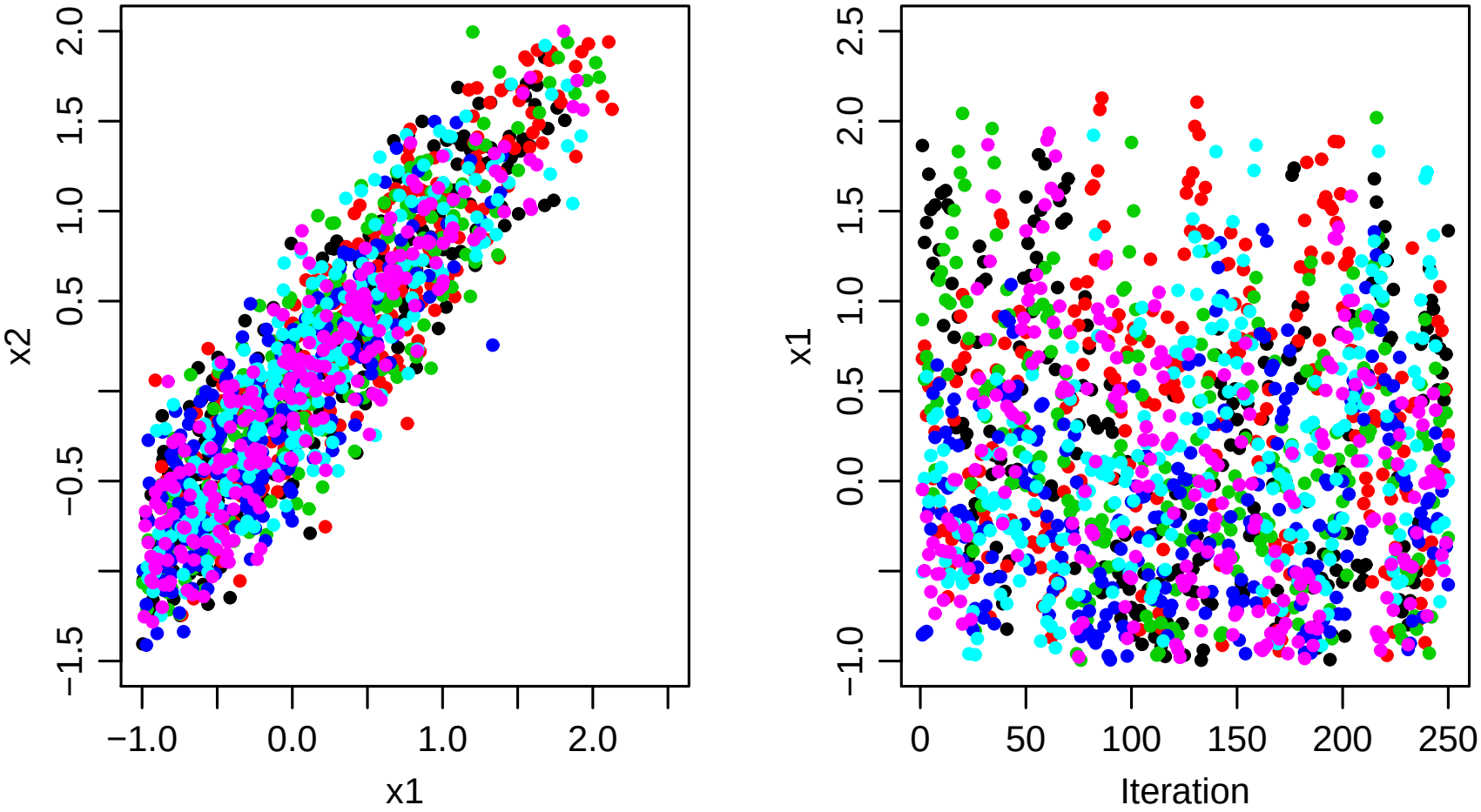
Two papers from 2012 hint at how MCMC might not require any (or very minimal) random inputs:

> Murray and Elliott, "Driving Markov chain Monte Carlo with a dependent random stream".
>
> Neal, "How to view an MCMC simulation as a permutation, with applications to parallel simulation and improved importance sampling".

They describe closely related methods, but differ in motivations. Murray and Elliott consider how to handle low quality "random" numbers. One application I consider is parallel simulation with all threads using the same random number stream (but hopefully giving different results).

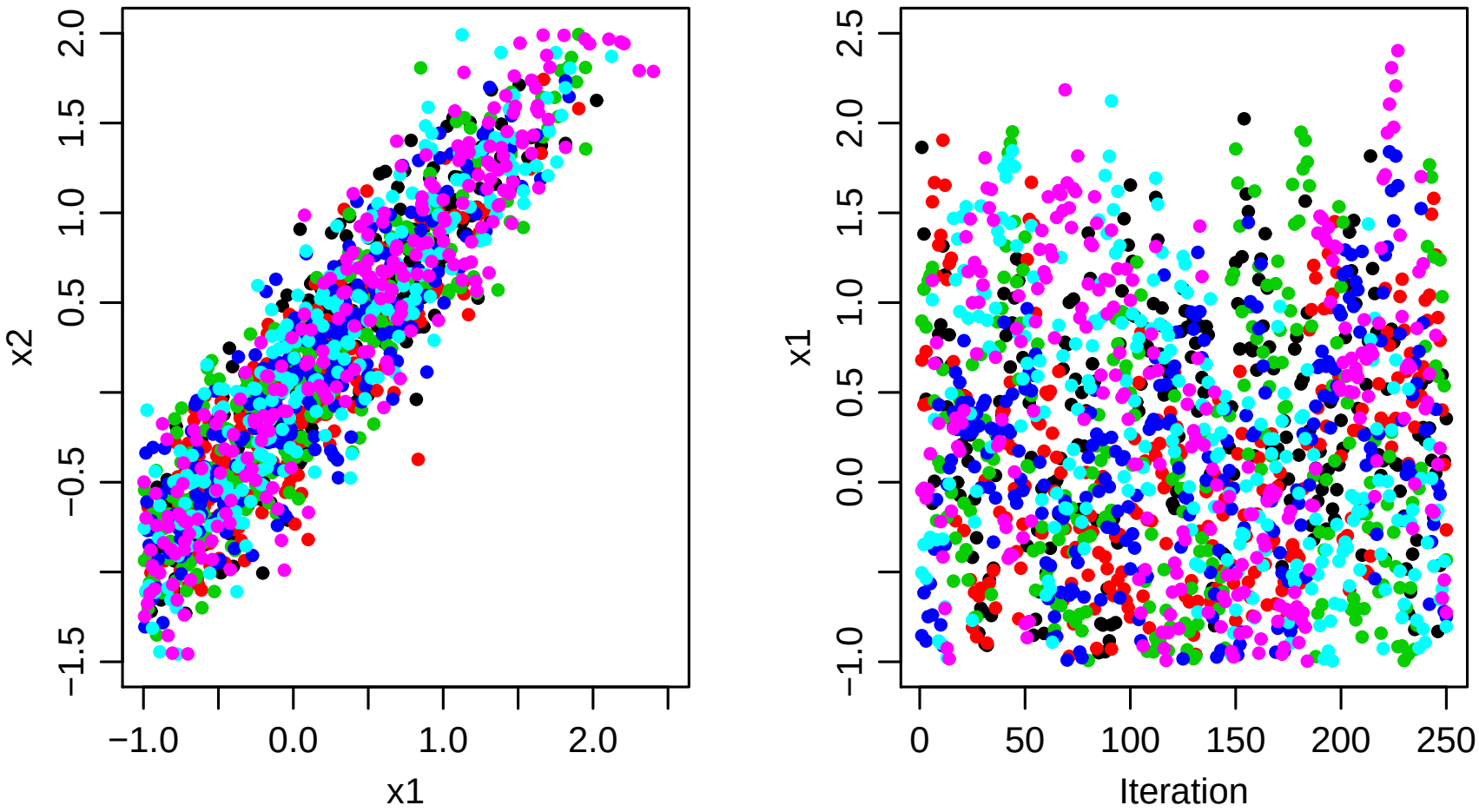
A simple version for Gibbs sampling: Include a uniform value, u, in the state space. To sample a new value for component x from its conditional distribution with CDF F, set $x_{t+1} = F^{-1}(u_t)$, and set $u_{t+1} = F(x_t)$ (so the method is reversible). Also add some c_t, to u_t, wrapping around to stay in $[0, 1]$.

Gibbs Sampling for a Truncated Bivariate Gaussian



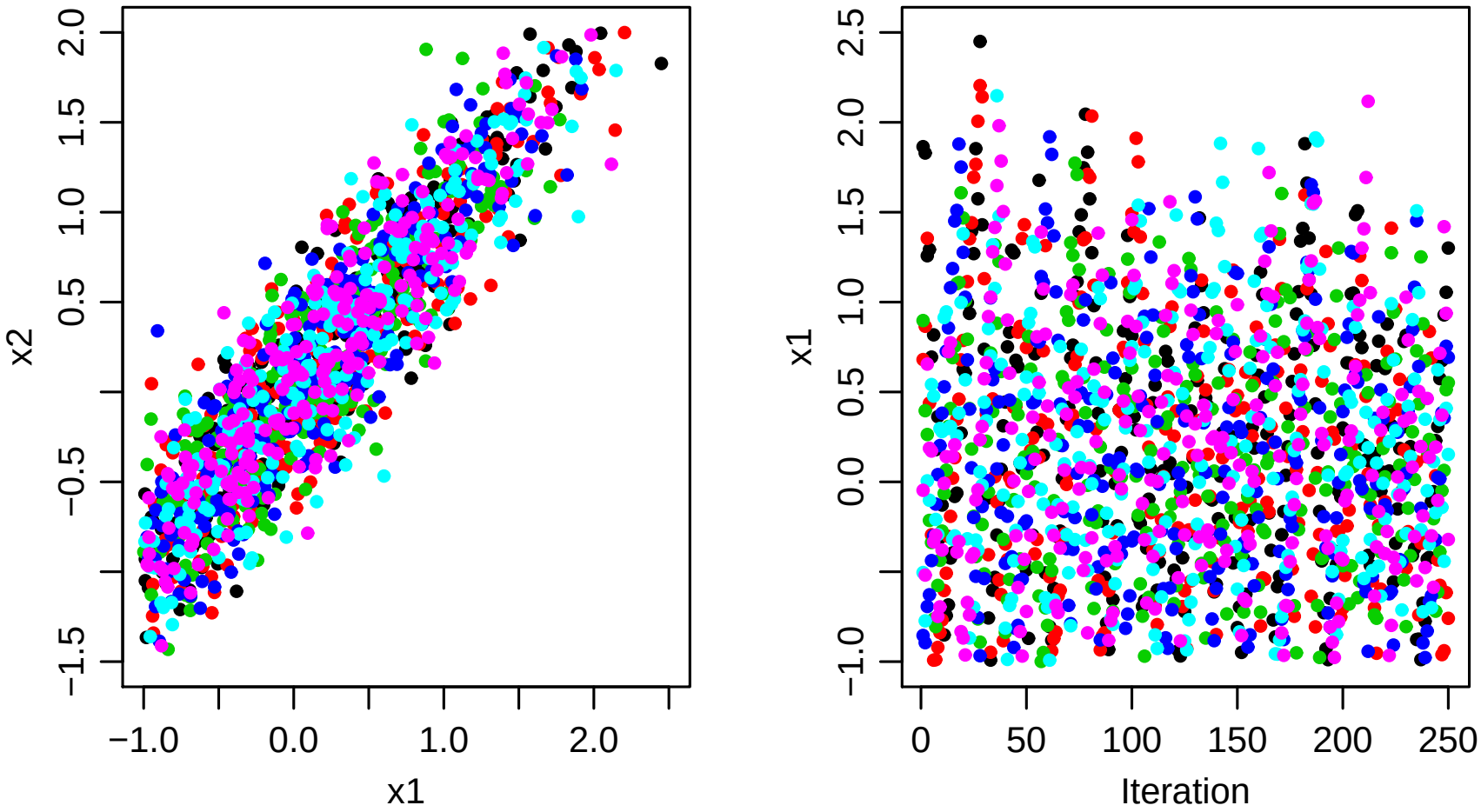
Six independent chains (different colours) sampling a bivariate Gaussian with means of zero, standard deviations of one, and correlation 0.95, truncated to the interval $(-1, 2.5)$ for x_1 and the interval $(-1.5, 2)$ for x_2.

Deterministic Gibbs Sampling — Random c_t



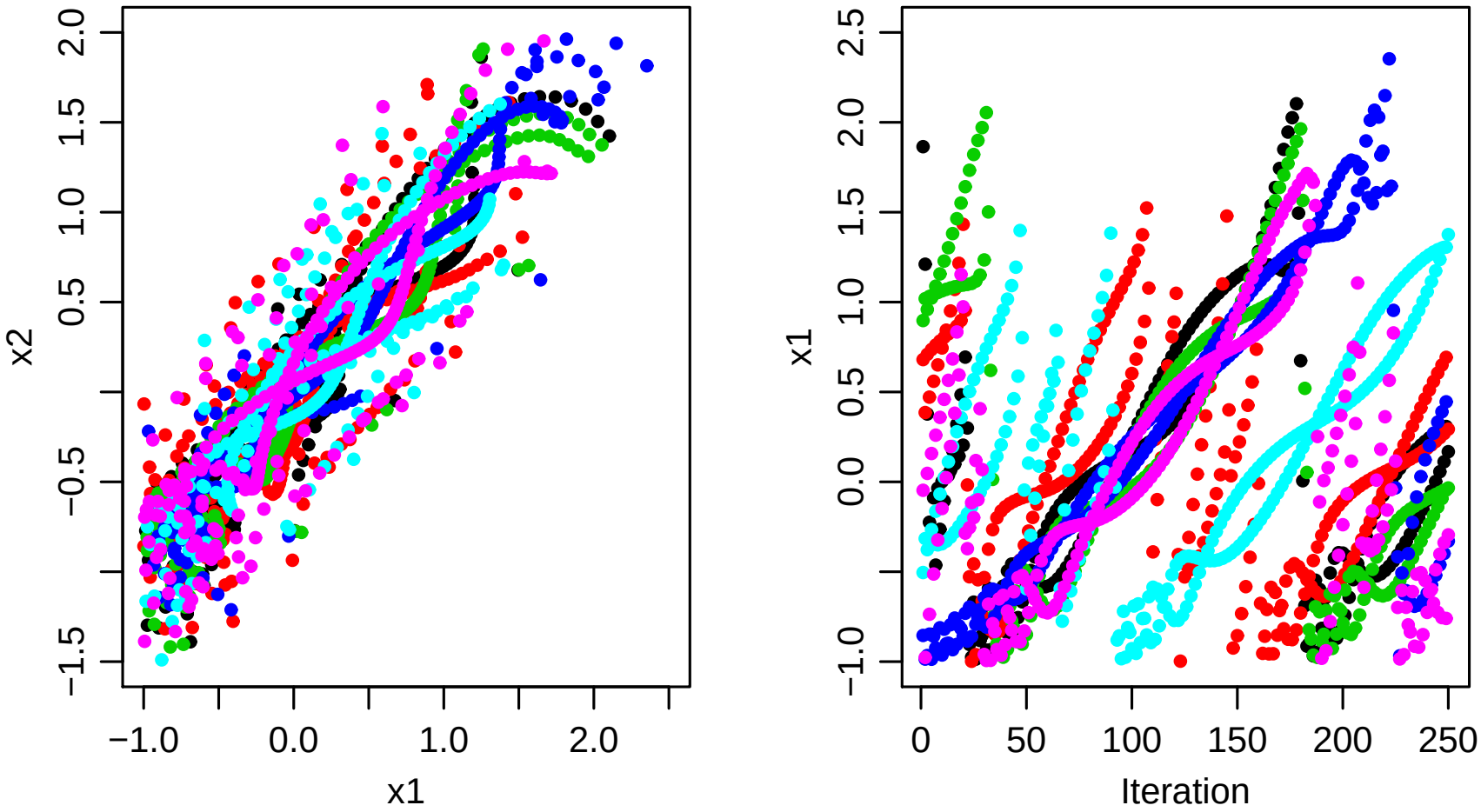
The six chains have different, random initial u values, but are thereafter updated by the same mapping (same random c_t for all chains).

Deterministic Gibbs Sampling — All $c_t = 0.211$



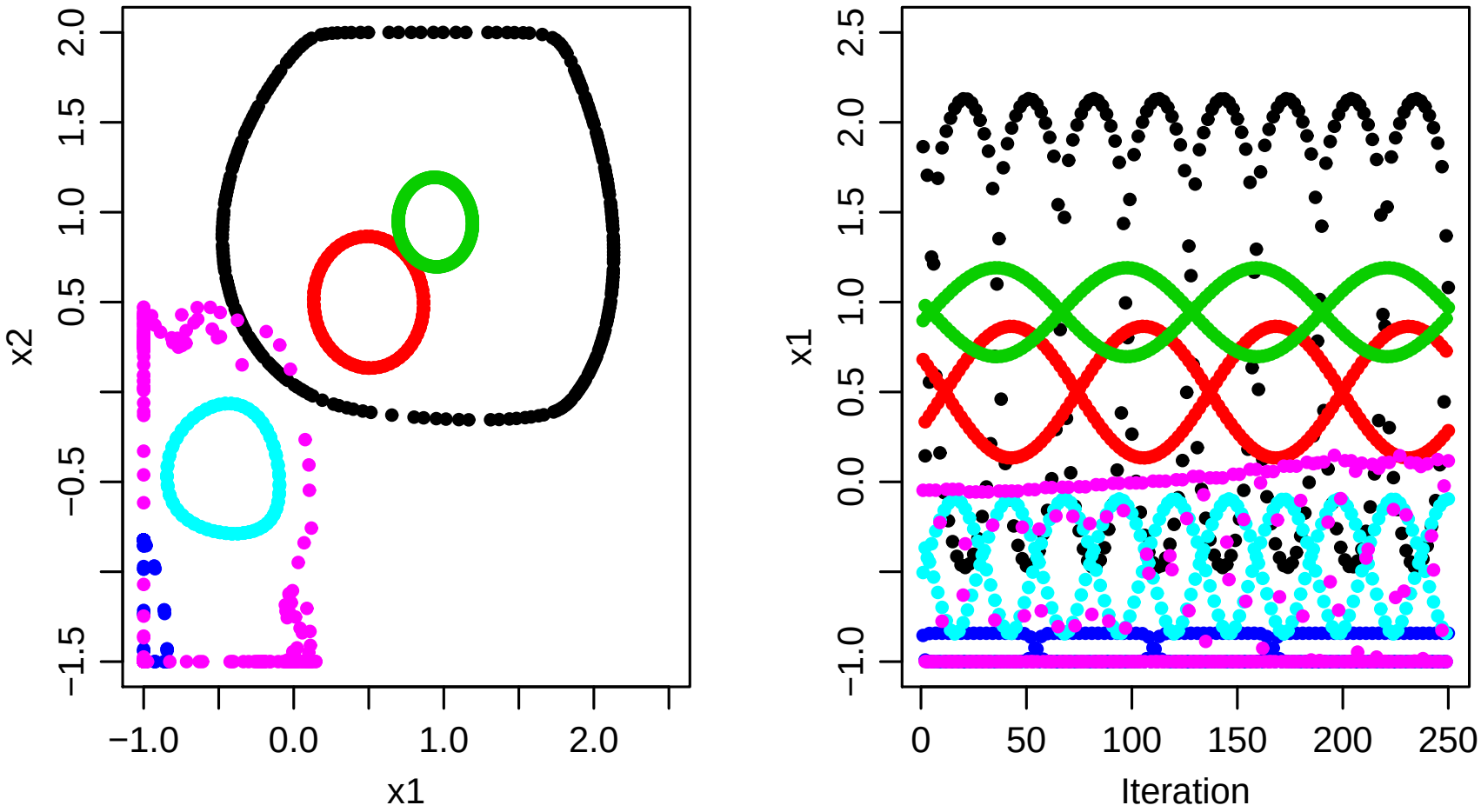
Maybe better than standard Gibbs sampling?

Deterministic Gibbs Sampling — All $c_t = 0.017$



Very pretty. Maybe even correct...

Deterministic Gibbs Sampling — All $c_t = 0$



Obviously gone a bit too far here...

A Little Paper from 1991

Horowitz, "A Generalized Guided Monte Carlo Algorithm".

> Considers Hamiltonian Monte Carlo with one-step trajectories (also called Langevin Monte Carlo) — but with a trick so that these steps nevertheless usually keep going in the same direction (except on a rejection).
>
> This is an early example of a "non-reversible" Markov chain Monte Carlo method.

Using one-step trajectories would sometimes be helpful because other updates (eg, of hyperparameers or discrete variables) could be done more often.

Unfortunately, even with the non-reversibility trick, Horowitz's method is not as efficient as HMC with long trajectories — keeping reversals rare requires a low rejection rate, only obtainable inefficiently by taking small steps.

The Non-Reversibility Trick

Do each update of (x, p) as three steps:

1) Update p to $\alpha p + \sqrt{1-\alpha^2}\, n$, where α is slightly less than 1 and n is a $N(0, \sigma^2 I)$ random variable (assuming that $K(p) = |p|^2/2\sigma^2$).

2) Propose a new state by doing one leapfrog step for (x, p) and then negating p. Accept or reject this proposal the usual way.

3) Negate p.

All steps leave the desired distribution invariant and are reversible.

Their sequential combination leaves the desired distribution invariant but is not reversible.

For α near 1, Step (1) only slightly changes p. If Step (2) accepts, the negation in the proposal is canceled by the negation in Step (3). But a rejection will reverse p, leading the chain to almost double back on itself.

Non-Reversible Modification of the Accept/Reject Decision

To decide whether to accept a Metropolis proposal to move from x to x^*, one can check whether $u < \pi(x^*)/\pi(x)$, with u a random uniform over $[0, 1]$.

Equivalently, one can check whether $\pi(x^*) > s$, where s is a random uniform over $[0, \pi(x)]$.

Rather than choosing s randomly, we can make it (or $u = s/\pi(x)$) part of the state, and update it in any way that leaves the joint distribution invariant.

One possible update: For some constant δ, add/subtract δs to s, reflecting off the boundaries at 0 and $\pi(x)$.

It may be wise to also randomly alter s by a small amount, to ensure that it will vary over its whole range.

Benefit for High-Dimensional Metropolis Updates

The number of simple Metropolis updates needed to reach an independent point increases in proportion to dimensionality, D, if we assume (as is typical) that the range of variation in the log density increases as $\sqrt{D}$ (Caracciolo, Pelisseta, and Sokal 1994).

This is a consequence of an update typically changing the log density by ≈ 1, and doing so in a random walk.

We might hope that the non-reversible modification of accept/reject decisions could avoid this — when s is small, rejections are less likely, and one would expect a drift to regions of small log density, with the opposite when s is large. Since s will change only slowly, this should reduce random walk behaviour.

I've found in tests on D-dimensional Gaussian distributions with identity covariance that there is a gain — but only of about 10% (empirically, the percentage gain is constant as D increases).

Improving Horowitz's Method

We might also hope that non-reversible accept/reject decisions could benefit Horowitz's one-step HMC method.

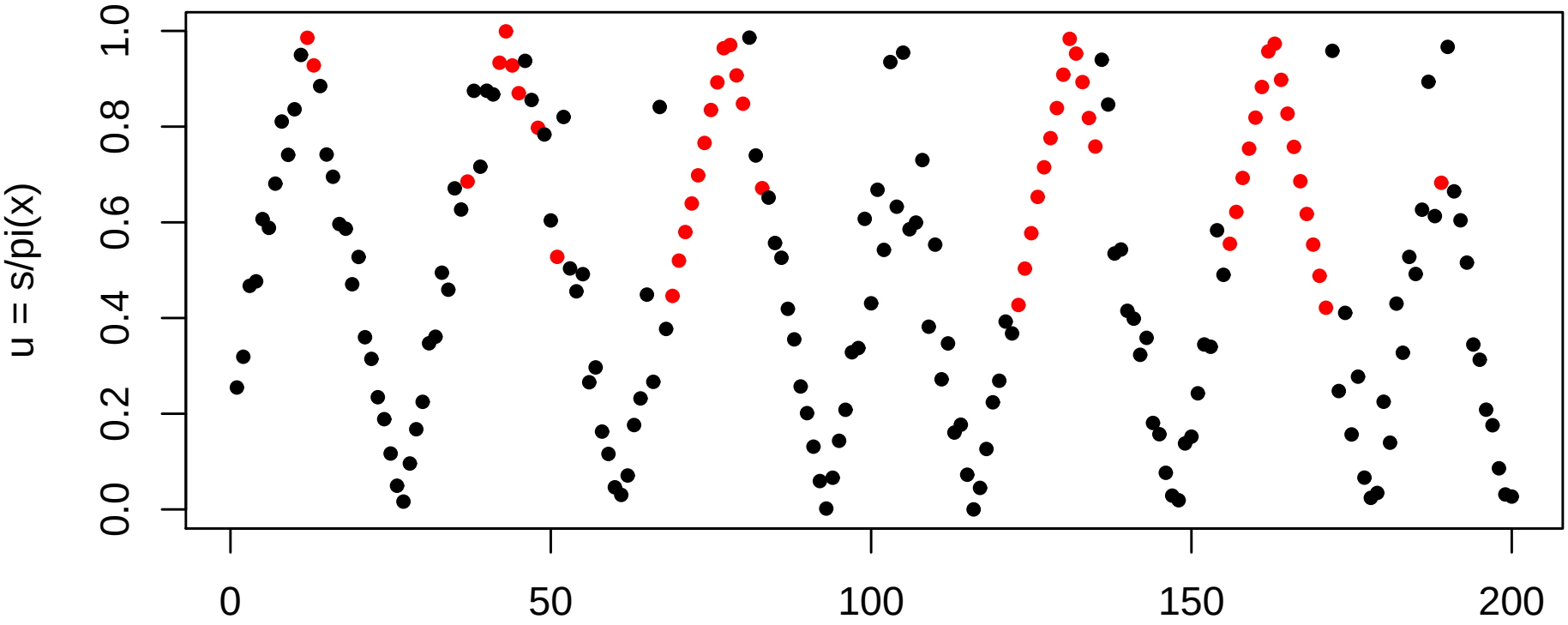
The problem is that rejections cause reversals of direction, so a small step size is needed to reduce rejection rate. But a higher rejection rate would be tolerable if rejections cluster in time, producing long runs of no rejections.

Note: Rejection-free runs of 20, 0, 20, 0, ... are better than rejection-free runs of 10, 10, 10, 10, ..., since N of the former will move a distance proportional to $20\sqrt{N/2} \approx 14\sqrt{N}$, but N of the latter only $10\sqrt{N}$.

The Improved Method for a Gaussian Distribution

I tried this for a bivariate Gaussian with variances of 1 and correlation 0.98.

I used $\eta = 0.21$, $\alpha = 0.97$, and $\delta = 0.067$. Here are the values for $u = s/\pi(x)$ for 200 consecutive iterations, with rejections in red:



Autocorrelation time for a coordinate was 17.2, compared to 25.9 without the non-reversible update of s.

The Improved Method for a Neural Network Model

I've also tried the improved method for a simple Bayesian neural network model (binary classification, 2 inputs, 12 hidden units, 300 training cases).

Preliminary results show that one-step HMC can almost match performance of multi-step HMC with hyperparameters fixed, and may be better when hyperparameters are sampled via separate Gibbs sampling steps.

References

Alder, B. J. and Wainwright, T. E. (1959) "Studies in molecular dynamics. I. General method", *Journal of Chemical Physics*, vol. 31, pp. 459-466.

Caracciolo, S, Pelisseto, A, and Sokal, A. D. (1994) "A general limitation on Monte Carlo algorithms of Metropolis type", *Physical Review Letters*, vol. 72, pp. 179-182. Also at `arxiv.org/abs/hep-lat/9307021`

Duane, S., Kennedy, A. D., Pendleton, B. J., and Roweth, D. (1987) "Hybrid Monte Carlo", *Physics Letters B*, vol. 195, pp. 216-222.

Gal, Y. and Ghahramani, Z. (2016) "Dropout as a Bayesian approximation: Representing model uncertainty in deep learning", `arxiv.org/abs/1506.02142`.

Gelfand, A. E. and Smith, A. F. M. (1990) "Sampling-based approaches to calculating marginal densities", *Journal of the American Statistical Association*, vol. 85, pp. 398-409.

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. R. (2012) "Improving neural networks by preventing co-adaptation of feature detectors", `arxiv.org/abs/1207.0580`

Hinton, G. E. and Sejnowski, T. J. (1986) "Learning and relearning in Boltzmann machines", in Rumelhart, D. E., McClelland, J. L., and the PDP Research Group, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 1: Foundations*, Cambridge, Massachusetts: MIT Press.

Horowitz, A. M. (1991) "A generalized guided Monte Carlo algorithm", *Physics Letters B*, vol. 268, pp. 247-252.

MacKay, D. J. C. (1992) "A practical Bayesian framework for backpropagation networks", *Neural Computation*, vol. 4, pp. 448-472.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E. (1953) "Equation of state calculations by fast computing machines", *Journal of Chemical Physics*, vol. 21, pp. 1087-1092.

Murray, I. and Elliott, L. T. (2012) "Driving Markov chain Monte Carlo with a dependent random stream", `arxiv.org/abs/1204.3187`

Neal, R. .M. (1994) *Bayesian Learning for Neural Networks*, PhD thesis, University of Toronto.

Neal, R. M. and Zhang, J. (2006) "High dimensional classification with Bayesian neural networks and Dirichlet diffusion trees", in I. Guyon, S. Gunn, M. Nikravesh, and L. A. Zadeh (editors) *Feature Extraction: Foundations and Applications*, Studies in Fuzziness and Soft Computing, Volume 207, Springer, pp. 265-295.

Neal, R. M. (2012) "How to view an MCMC simulation as a permutation, with applications to parallel simulation and improved importance sampling", `arxiv.org/abs/1205.0070`

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986) "Learning internal representations by error propagation", in Rumelhart, D. E., McClelland, J. L., and the PDP Research Group, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 1: Foundations*, Cambridge, Massachusetts: MIT Press.