Presentation Sketch Script

B. Han, T.M. Hodgson, M. Holden & M. Puza

March 13, 2019

1 Introduction

"Recent Advances in Langevin Monte Carlo"
The aim of MCMC is to be able to accurately sample from an unknown probability distribution. When do we not know a distribution?

— Bayesian Statistics, only have some data and an incomplete idea of what the distribution looks like

— Molecular Dynamics

All MCMC methods aim to sample as accurately and efficiently as possible. We will focus on distributions of the form

$$\pi(x) = \mathcal{Z}^{-1}\mathrm{e}^{-U(x)}$$

This distribution goes by many names as it crops up in many areas:

— Statisticians know it as a 'log-linear model'

— Mathematicians often call it a 'Gibbs measure'

— Physicists know it as the 'Boltzmann distribution' or 'canonical ensemble'

In fact, the Hammersley-Clifford theorem states that any measure that satisfies a Markov property can be expressed as a Gibbs measure. So although we only look at distributions of this form, it is not actually as much of a restriction as it initially appears. Borrowing the terminology from MD, we call U the potential function ($\mathcal{Z}$ is a normalising constant). We know potential, not normaliser. Let's look pursue this molecular dynamics link to gain some intuition. How does this distribution arise?

In MD, U describes the potential energy of a particle, for this reason we call U a potential well. (Picture of potential well). If we let a particle move in this well, this is what we see (animation?). The particle moves in the well, up and down each side.

The equation of motion for a particle moving like this (Brownian dynamics) is the SDE

$$\mathrm{d}X_t = -\nabla U(X_t)\,\mathrm{d}t + \sqrt{2}\,\mathrm{d}W_t$$

the (overdamped) Langevin equation. A natural question to ask is where is this particle on average? It spends most of the time in the bottom of the well (can see that just by watching), sometimes makes it quite high up the sides. Shock horror, it is exactly given by π! This is what we call the invariant measure of the SDE (kind of).

2 Algorithms

So back to statistics: the problem of statistical sampling has just been reduced to watching particles in wells! Bonzer! ... Except not quite. Just like in the ordinary differential equation case, one can't solve exactly in many cases, so we must discretise. Like in ODEs we use the Euler method!

Applying this to the Langevin equation gives an iterative scheme to approximate the continuous diffusion (this is the MC from which MCMC gets its name).

$$X_{n+1} = X_n - h\nabla U(X_n) + \sqrt{2h}Z_{n+1}$$

Let's see what this looks like (trace plot of approximation? Or vis in well with a higher stepsize) This is only exact when the step size tends to zero, which we obviously can't do. This means that the invariant measure

(the average positions of the particles) may not be the same as the continuous case. Or we may not even be able to find one! (Chain is transient/not ergodic)

Example:

Most extreme case: $h = 1, U(x) = x^2/2$. Obvious invariant measure is $N(0,1)$, but chain (immediately) converges to $N(0,2)$.

So we have to be careful with stepsize. Also if the potential is superlinear (sides of the well are too steep), the iterative scheme diverges, even though the underlying diffusion is ergodic.

Can we fix this? Yeeees, with Metropolis rejection (explain? Idea is to just reject some of the samples with some probability - show vis with rejects). Doing so means that whenever the underlying diffusion is exp ergodic?, the chain also will be (in the limit – not actually useful - show vis.). Still runs in to problems when the potential is superlinear.

So this is the direction that most research has taken since its initial proposal in 1996. We have been looking at a different method that avoids Metropolis rejection – TAMING!

3 Taming the Beast

We have seen in the visualisation that `ULA` and `MALA` don't quite work. The issue is the gradient of the potential well. Taming swaps the gradient for a function that is 'close' to the gradient, but preserves the invariant measure of the diffusion.

$$\mathrm{d}X_t = -\nabla U(X_t)\,\mathrm{d}t + \sqrt{2}\,\mathrm{d}W_t \implies \mathrm{d}X_t = -T(X_t)\,\mathrm{d}t + \sqrt{2}\,\mathrm{d}W_t$$

Different choices of T are available, we consider

$$T = \frac{\nabla U}{1 + h\nabla U}$$

What does this do? It reduces the effect the gradient has on the iteration. Visualise.

Stiffness effects motivates coordinatewise? Show how taming deals with stiff problems in 2d. (Impacts ability in ill-conditioned problems).

One final modification we wish to mention is to use a different method of discretisation: Higher order (Stochastic Runge-Kutta) `tHOLA`, Leimkuhler-Matthews (Non-Markovian) `LM`. We don't have time to go into these methods here but ask at the end if you're interested in how they work.

4 Horse Racing

So we have given a heuristic (qualitative) motivation of how these algorithms are different. Can we quantify each methods effectiveness on different potentials/distributions?

The existing results on `tULA` only looked at first and second moments Brosse et al.. This is sufficient only for Gaussian distributions, which aren't even superlinear. We wish to get some stronger results in terms of metrics.

Shitty boxplots $\longrightarrow$ Histograms? Wasserstein? KL? Computer time? Pick two or three plots that best illustrate errors, or use vis.

We also have some theoretical non-asymptotic error bounds in these metrics.

5 Conclusion

What algorithm is best? Probably problem dependent, depends what you want out of the alorithm. Taming will give you an answer, which may be inaccurate – but at least it is an answer.

Next Steps

There are many many methods we haven't considered: HMC, mMALA, SGLD. Discuss SGLD? So hot right now NeurIPS.

Future Work

We'd like to test these methods on real data, or even just simulated data, when we don't know the gradient analytically. This is much more relevant for the end user than a theoretical bound on convergence. Especially wish to test on 'big data' as many (all?) MCMC methods suffer the curse of dimensionality. Lower dimensions

may not show off the benefits of taming, may only become relevant in high dimensions. Also for algorithms with very complex iteration steps, they may lend themselves to parallelisation better than simpler algorithms.

Finally, we've made all the code used for testing available on github, as well as the visualisation.

Any questions? PLease say no