Principled subsampling and super-efficiency for Bayesian inference

Gareth Roberts, University of Warwick

Joint work with Paul Fearnhead, Joris Bierkens, Murray Pollock, Andi Wang, Adam Johansen, and others ...

Warwick Statistics

i-like.org.uk

CoSInES

Talk outline

- The zig-zag as an alternative to MCMC. "The Zig-Zag Process and Super-Efficient Sampling for Bayesian Analysis of Big Data" arXiv:1607.03188, Ann Stat 2019
- Quasi-stationary Monte Carlo and the ScaLE algorithm "The scalable Langevin exact algorithm: Bayesian inference for big data" arXiv:1609.03436

Both these ideas are examples of Continuous-time Monte Carlo algorithms.

Talk outline

- The zig-zag as an alternative to MCMC. "The Zig-Zag Process and Super-Efficient Sampling for Bayesian Analysis of Big Data" arXiv:1607.03188, Ann Stat 2019
- Quasi-stationary Monte Carlo and the ScaLE algorithm "The scalable Langevin exact algorithm: Bayesian inference for big data" arXiv:1609.03436

Both these ideas are examples of Continuous-time Monte Carlo algorithms.

Super-Efficiency:

$$\frac{\text{computational cost of running algorithm}}{\text{cost of one single likelihood evaluation}} \longrightarrow 0$$

in the big data asymptotic.

Likelihood intractability due to data size

MCMC is the workhorse of Bayesian inference. Since it requires large numbers of realisations of the posterior density $\pi(x)$, it relies on these evaluations to be quick. However connsider (for example) the big (tall) data case:

$$\pi(x) = \prod_{i=1}^{n} \pi_i(x)$$

Evaluation of $\pi(x)$ is typically an $O(n)$ calculation.

Likelihood intractability due to data size

MCMC is the workhorse of Bayesian inference. Since it requires large numbers of realisations of the posterior density $\pi(x)$, it relies on these evaluations to be quick. However connsider (for example) the big (tall) data case:

$$\pi(x) = \prod_{i=1}^{n} \pi_i(x)$$

Evaluation of $\pi(x)$ is typically an $O(n)$ calculation.

Does that mean that exact Bayesian inference is not realistically possible for huge data sets?

Likelihood intractability due to data size

MCMC is the workhorse of Bayesian inference. Since it requires large numbers of realisations of the posterior density $\pi(x)$, it relies on these evaluations to be quick. However connsider (for example) the big (tall) data case:

$$\pi(x) = \prod_{i=1}^{n} \pi_i(x)$$

Evaluation of $\pi(x)$ is typically an $O(n)$ calculation.

Does that mean that exact Bayesian inference is not realistically possible for huge data sets?

Maybe we can get away with just computing some of π at each step? This is known as a subsampling approach.

Piecewise-deterministic Markov processes

Continuous time stochastic process, denote by Z_t.

The dynamics of the PDP involves random events, with deterministic dynamics between events and possibly random transitions at events.

(i) The deterministic dynamics. eg specified through an ODE

$$\frac{\mathrm{d}z_t}{\mathrm{d}t} = \Phi(z_t), \tag{1}$$

So

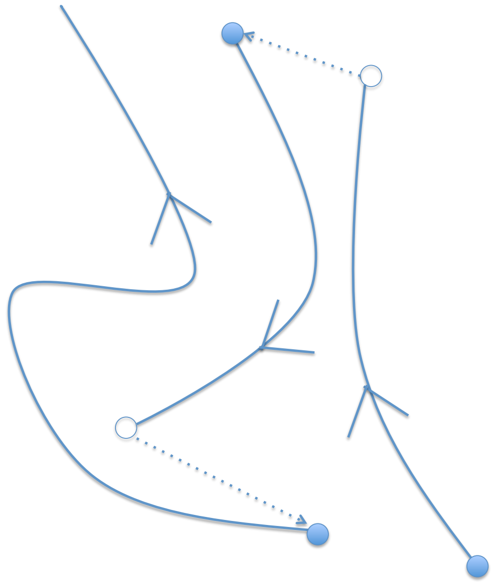
$$z_{s+t} = \Psi(z_t, s)$$

for some function Ψ.

(ii) The event rate. Events occur at rate, $\lambda(z_t)$,

(iii) The transition distribution at events. At each event time τ, Z changes according to some transition kernel

PDMP



PDMP

Date back to 1951 paper by Mark Kac on the telegraph process.

Mathematical foundations: Davis (1984, JRSS B)

Non-reversible Markov process.

Intrinsically continuous in time unlike (almost all) algorithms. Why would they ever be useful for simulation?

Unlike diffusion processes they are comparatively understudied, and underused (either for models or in stochastic simulation).

PDMP

Date back to 1951 paper by Mark Kac on the telegraph process.

Mathematical foundations: Davis (1984, JRSS B)

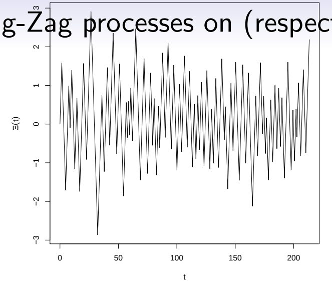
Non-reversible Markov process.

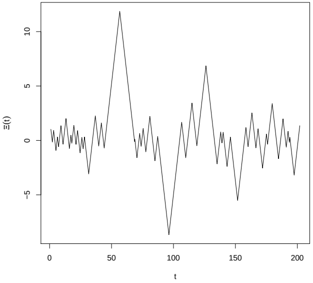
Intrinsically continuous in time unlike (almost all) algorithms. Why would they ever be useful for simulation?

Unlike diffusion processes they are comparatively understudied, and underused (either for models or in stochastic simulation).

.... until recently

One-dimensional Zig-Zag processes on (respectively) Gaussian and Cauchy targets.





Canonical Zig-Zag

State (X_t, V_t) in dimension d.

$$dX_t = V_t \ dt$$

$V_{t-}^{(i)} \to 1 - V_{t-}^{(i)}$ at rate

$$\lambda_i(X_t, V_t) = \lambda_i^0(X_t, V_t) \equiv \max\left\{0, -V_{t-}^{(i)} \frac{\partial \log \pi(X_{t-})}{\partial X^{(i)}}\right\}$$

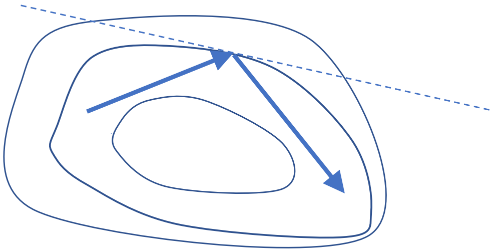
Invariant distribution is

$$\pi_E(x, v) \propto \pi(x)$$

ie in stationarity X and V are independent with V being uniform over all configerations: $(\pm 1, \pm 1, \ldots \pm 1)$

Alternatives to Zig-Zag

Closely related to another PDMP scheme, the bouncy particle sampler (BPS), [Bouchard-Côté et al., 2015].



Many alternatives/variants available.

Refreshment

Back to Zig-Zag

There is a lot more flexibility!

For instance, can take

$$\lambda_i(x, v) = \lambda_i^0(x, v) + \nu(x)$$

for any function ν.

Why might we do this?

Refreshment

Back to Zig-Zag

There is a lot more flexibility!

For instance, can take

$$\lambda_i(x, v) = \lambda_i^0(x, v) + \nu(x)$$

for any function ν.

Why might we do this?

To help visit different parts of the state space.

But the larger ν is, the *closer* to reversibility.

The canonical Zig-Zag is the *most non-reversible*.

Implementation

How do we simulate continuous time stochastic process like this?

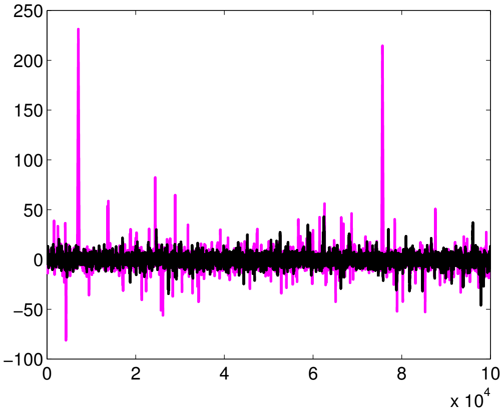
By using thinned poisson processes

For example, if $|(\log \pi)'(x)| < c$, simulate a Poisson process of rate c (by simulating the exponential inter-arrival times). Then at each poisson time, we accept as a direction change with probability $\max(-(\log \pi)'(x), 0)/c$.

This makes the algorithm inexpensive to implement as we only need to calculate $(\log \pi)'(x)$ occasionally.

There are many other details .... though the method is not so complicated.

Zig zag process for sampling the Cauchy distribution



$T = 10,000$

Multi-dimensional zig zag process

Multi-dimensional zig zag process: here we have a multi-dimensional binary velocity, eg $(1, -1, -1, 1, 1, -1, 1, 1)$.

Efficient sampling (currently for potentials with locally Lipschitz gradients in multiple dimensions, but with obvious ways to extend).

Subsampling

Motivation: intractable likelihood problems where calculating π at any one fixed location is prohibitively expensive (given that very many evaluations will be required to run the algorithm. For this talk, concentrate on the Bayesian setting:

$$\pi(x) = \prod_{i=1}^{N} \pi_i(x)$$

Eg we have N observations (but this method is not in any way restricted to the independent data case).

Subsampling

Motivation: intractable likelihood problems where calculating π at any one fixed location is prohibitively expensive (given that very many evaluations will be required to run the algorithm. For this talk, concentrate on the Bayesian setting:

$$\pi(x) = \prod_{i=1}^{N} \pi_i(x)$$

Eg we have N observations (but this method is not in any way restricted to the independent data case).

Aim to be lazy and only use a small number of the terms in the product.

Subsampling

Motivation: intractable likelihood problems where calculating π at any one fixed location is prohibitively expensive (given that very many evaluations will be required to run the algorithm. For this talk, concentrate on the Bayesian setting:

$$\pi(x) = \prod_{i=1}^{N} \pi_i(x)$$

Eg we have N observations (but this method is not in any way restricted to the independent data case).

Aim to be lazy and only use a small number of the terms in the product.

For instance we might try pseudo-marginal MCMC (Beaumont, 2003, Andrieu and Roberts, 2009).But that would require an unbiased non-negative estimate of $\pi(x)$ with variance which is stable as a function of N.

Subsampling

Motivation: intractable likelihood problems where calculating π at any one fixed location is prohibitively expensive (given that very many evaluations will be required to run the algorithm. For this talk, concentrate on the Bayesian setting:

$$\pi(x) = \prod_{i=1}^{N} \pi_i(x)$$

Eg we have N observations (but this method is not in any way restricted to the independent data case).

Aim to be lazy and only use a small number of the terms in the product.

For instance we might try pseudo-marginal MCMC (Beaumont, 2003, Andrieu and Roberts, 2009).But that would require an unbiased non-negative estimate of $\pi(x)$ with variance which is stable as a function of N. But this is not possible for a product without computing cost which is at least $O(N)$.

Subsampling within PDMP

PDMP for the exploration of high-dimensional distributions (such as zig-zag or the ScaLE algorithm, Fearnhead, Johansen, Pollock and Roberts, 2016) typically use $\log \pi(x)$ rather than $\pi(x)$ and

$$\log \pi(x) = \sum_{i=1}^{N} \log \pi_i(x)$$

for which there are well-behaved $O(1)$ cost, $O(1)$ variance (or sometime a little worse). Can we use this?

Zig zag switching rate $\max\left(0, -j \sum_{i=1}^{N} (\log \pi)'_i(x)\right) \rightsquigarrow O(N)$ calculation at every switch

Subsampling for zig-zag

Sub-sampling

- Determine global upper bound M for switching rate
- Simulate Exponential(M) random variable T
- Generate $I \sim \text{discrete}(\{1,\ldots,N\})$
- Accept the generated T as a "switching time" with probability $N \max\left(0, -j(\log \pi_I)'(Y(T))\right)/M$

Theorem: This works! (invariant distribution π)

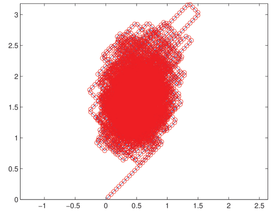
Subsampling + control variates

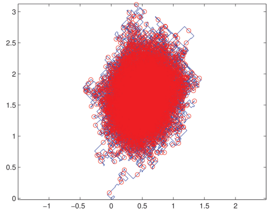
Crudely, for an $O(1)$ update in state space:

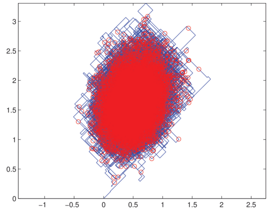
- Without subsampling, $O(N)$ computations required
- Using subsampling, gain factor $N^{1/2}$ $\rightsquigarrow$ complexity $O(N^{1/2})$ per step
- Using control variates, gain additional factor $N^{1/2}$ $\rightsquigarrow$ complexity $O(1)$ per step

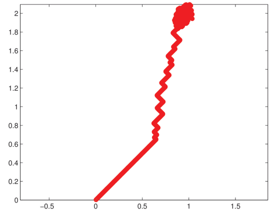
Superefficiency We call an epoch the time taken to make one function evaluation of the target density π. The control variate subsampled zig-zag is superefficient in the sense that the effective sample size from running the algorithm per epoch diverges.

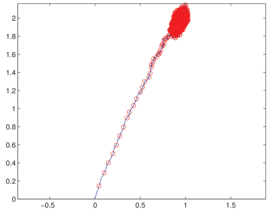
Subsampling + control variates – Logistic growth

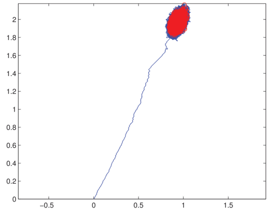

(a) $N = 100$


(b) $N = 100$


(c) $N = 100$


(d) $N = 10,000$

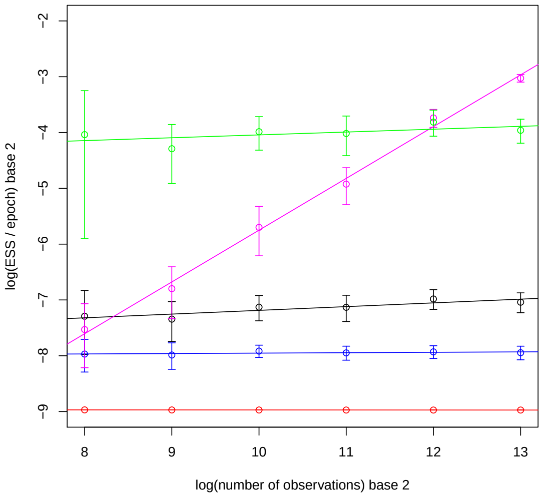

(e) $N = 10,000$


(f) $N = 10,000$

[Bierkens, Roberts, 2015, `http://arxiv.org/abs/1509.00302`]

[Bierkens, Fearnhead, Roberts, Ann Stat to appear]

Effective Sample Size per epoch



Summary of Zig-Zag

- PDMPs have many uses for simulation of stochastic processes (even those very different from PDMPs) as well as steady state simulation.

Summary of Zig-Zag

- PDMPs have many uses for simulation of stochastic processes (even those very different from PDMPs) as well as steady state simulation.
- Subsampling and control-variate tweaks greatly improve efficiency in certain situations. PDMP are particularly amenable to this.

Summary of Zig-Zag

- PDMPs have many uses for simulation of stochastic processes (even those very different from PDMPs) as well as steady state simulation.
- Subsampling and control-variate tweaks greatly improve efficiency in certain situations. PDMP are particularly amenable to this.
- More work is needed on studying the theoretical and empirical properties of these algorithms, and exploiting their flexibility. (Though lots more I have not told you ...)

Summary of Zig-Zag

- PDMPs have many uses for simulation of stochastic processes (even those very different from PDMPs) as well as steady state simulation.
- Subsampling and control-variate tweaks greatly improve efficiency in certain situations. PDMP are particularly amenable to this.
- More work is needed on studying the theoretical and empirical properties of these algorithms, and exploiting their flexibility. (Though lots more I have not told you ...)
- Zigzag is a flexible and usually easy-to-implement method for simulating from a target distribution.

Summary of Zig-Zag

- PDMPs have many uses for simulation of stochastic processes (even those very different from PDMPs) as well as steady state simulation.
- Subsampling and control-variate tweaks greatly improve efficiency in certain situations. PDMP are particularly amenable to this.
- More work is needed on studying the theoretical and empirical properties of these algorithms, and exploiting their flexibility. (Though lots more I have not told you ...)
- Zigzag is a flexible and usually easy-to-implement method for simulating from a target distribution.
- Can zigzag be a competitor to Hamiltonian MCMC?

Quasi-stationary Monte Carlo

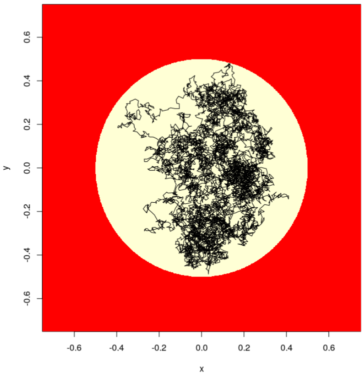
Traditional Markov chain Monte Carlo rests on the construction of an ergodic Markov chain designed to have a prescribed stationary distribution π.

Quasi-stationary Monte Carlo instead makes use of the conditional distribution of an killed stochastic process conditioned on not being killed.

This turns out to be a natural framework for subsampling without approximation.

Quasi-stationarity: boundary killing

Ant on a volcanic island undergoing Brownian motion, killed at τ_∂ when it touches lava.



What can be said about $\mathbb{P}(X_t \in \cdot \,|\tau_\partial > t)$ for large t?

Quasi-stationarity: interior killing

Take a continuous-time Markov process on $\mathbb{R}^d$

$$(X_t, \quad t \geq 0).$$

We then augment this process with an inhomogeneous Poisson process:

$$\tau_\partial := \inf \left\{ t \geq 0 : \int_0^t \kappa(X_s) \mathrm{d}s \geq \xi \right\},$$

where $\xi \sim \mathrm{Exp}(1)$, independent of X.

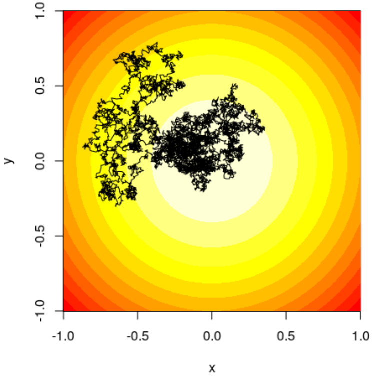
Here $\kappa : \mathbb{R}^2 \to [0, \infty)$ is a locally bounded function, the killing rate.

Quasi-stationarity: interior killing example

Take X to be a standard Brownian motion on $\mathbb{R}^2$, $\kappa(y) = \|y\|^2$.

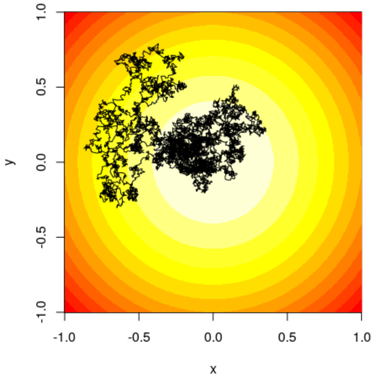
Quasi-stationarity: interior killing example

Take X to be a standard Brownian motion on $\mathbb{R}^2$, $\kappa(y) = \|y\|^2$.



Quasi-stationarity: interior killing example

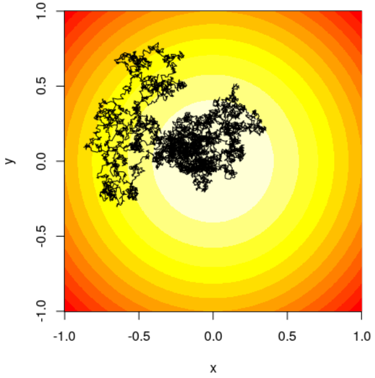
Take X to be a standard Brownian motion on $\mathbb{R}^2$, $\kappa(y) = \|y\|^2$.



What can be said about $\mathbb{P}(X_t \in \cdot \,|\tau_\partial > t)$ for large t?

Quasi-stationarity: interior killing example

Take X to be a standard Brownian motion on $\mathbb{R}^2$, $\kappa(y) = \|y\|^2$.



What can be said about $\mathbb{P}(X_t \in \cdot \,|\tau_\partial > t)$ for large t? Gaussian.

Quasi-stationarity

Say a probability measure μ is **quasi-stationary** if for any $t \geq 0$,

$$\mathbb{P}_\mu(X_t \in \cdot \,|\tau_\partial > t) = \mu(\cdot).$$

Quasi-stationarity

Say a probability measure μ is **quasi-stationary** if for any $t \geq 0$,

$$\mathbb{P}_\mu(X_t \in \cdot | \tau_\partial > t) = \mu(\cdot).$$

Say μ is **quasi-limiting** if for each measurable set E

$$\mathbb{P}_x(X_t \in E | \tau_\partial > t) \to \mu(E).$$

Quasi-stationarity

Say a probability measure μ is **quasi-stationary** if for any $t \geq 0$,

$$\mathbb{P}_\mu(X_t \in \cdot \,|\tau_\partial > t) = \mu(\cdot).$$

Say μ is **quasi-limiting** if for each measurable set E

$$\mathbb{P}_x(X_t \in E|\tau_\partial > t) \to \mu(E).$$

Rich literature in probability theory; e.g. population dynamics, and textbook of Collet et al (2013).

Quasi-stationarity

Say a probability measure μ is **quasi-stationary** if for any $t \geq 0$,

$$\mathbb{P}_\mu(X_t \in \cdot | \tau_\partial > t) = \mu(\cdot).$$

Say μ is **quasi-limiting** if for each measurable set E

$$\mathbb{P}_x(X_t \in E | \tau_\partial > t) \to \mu(E).$$

Rich literature in probability theory; e.g. population dynamics, and textbook of Collet et al (2013).

This actually arises quite naturally in computing:

$$\tau_\partial = \{\text{algorithm behaves very badly}\},$$

e.g. stack overflow, very slow runs, *cf. user-impatience bias*.

Characterisation of quasi-stationarity distributions

Discrete time

- Transition matrix P.
- π is stationary if
$$\pi P = \pi.$$
- π quasi-stationary if
$$\pi P = \lambda \pi,$$
some $0 < \lambda < 1$.
- Semigroup
$$\pi P^n = \lambda^n \pi.$$

Continuous time

- Rate matrix Q.
- π is stationary if
$$\pi Q = 0.$$
- π quasi-stationary if
$$\pi Q = -\lambda \pi,$$
some $\lambda > 0$.
- Semigroup
$$\pi P^t = e^{-\lambda t} \pi.$$

Characterisation of quasi-stationarity distributions

Discrete time

- Transition matrix P.
- π is stationary if
$$\pi P = \pi.$$
- π quasi-stationary if
$$\pi P = \lambda \pi,$$
some $0 < \lambda < 1$.
- Semigroup
$$\pi P^n = \lambda^n \pi.$$

Continuous time

- Rate matrix Q.
- π is stationary if
$$\pi Q = 0.$$
- π quasi-stationary if
$$\pi Q = -\lambda \pi,$$
some $\lambda > 0$.
- Semigroup
$$\pi P^t = e^{-\lambda t} \pi.$$

Theory of quasi-stationarity more delicate, so why bother ....

Quasi-stationary Monte Carlo

Start with a diffusion, for simplicity assume X is Brownian motion.

At time t, kill X at rate $\kappa(X_t)$.

Idea of quasi-stationary Monte Carlo: choose κ in such a way that the quasi-limiting distribution coincides with the target distribution π.

Quasi-stationary Monte Carlo

Start with a diffusion, for simplicity assume X is Brownian motion.

At time t, kill X at rate $\kappa(X_t)$.

Idea of quasi-stationary Monte Carlo: choose κ in such a way that the quasi-limiting distribution coincides with the target distribution π.

Need to take

$$\kappa(x) = \frac{1}{2}\frac{\Delta\pi(x)}{\pi(x)} + C$$

where Δ denotes the Laplacian and C is an arbitrary constant (which needs to be chosen so that κ is always non-negative.

How to extract samples from π?

For MCMC it is obvious to just take values of the chain after a while which should be close to samples from π.

It is clearly inefficient to take long runs of Brownian motion and just keep the ones which have not been killed.

How to extract samples from π?

For MCMC it is obvious to just take values of the chain after a while which should be close to samples from π.

It is clearly inefficient to take long runs of Brownian motion and just keep the ones which have not been killed.

Instead we have two approaches

- The Scalable Langevin Exact Algorithm: ScaLE which propagates a population of particles. Once one dies. it is resurrected from the location of one of the other particles.
- ReScaLE which uses a single trajectory which on death regenerates from a point along the trajectory to date.

Some implementational comments about ScaLE

The algorithm is implemented via an SMC framework.

In practice, we don't automatically kill particles and carry weighted particles instead in a more traditional SMC way. This is more efficient.

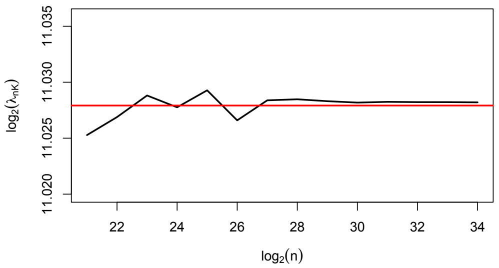
Crucial to efficiency is subsampling. Need tractability of f as well as a thinned Poisson process approach. All of this is analogous to the Zig-Zag set-up.

The underlying stochastic process is just Brownian motion. Simulation of Brownian motion is complicated by the need to simulate random variable such as BM's first exit time of a suitable hypercube. (Needed to ensure we can exactly implement the thinned PP method.

ScaLE

The key is that deciding on whether to kill a particle or not can be done using subsamples of the data set of size 2, with no loss of algorithmic efficiency.

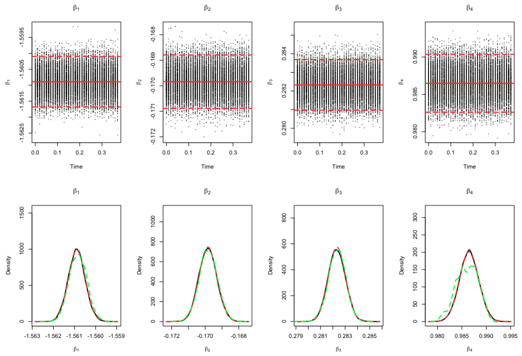
For example a logistic regression example using control variates:



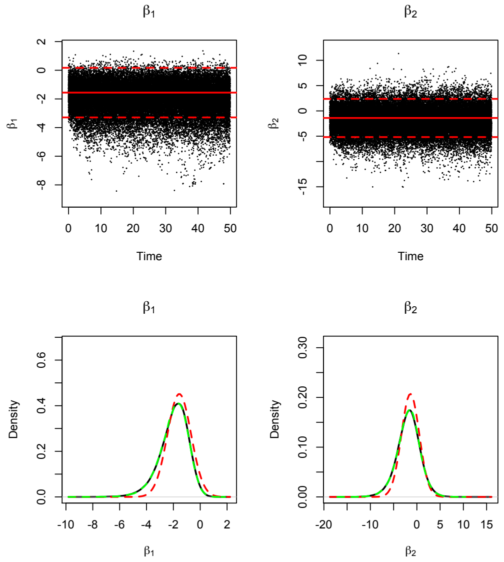
Logistic regression example

Airline data set (all flights in US over an extended period). Binary output of whether the flight was late.

49,665,450 individual records of the data set were accessed (equivalent to roughly 0.0029 full data evaluations) for the following output.



Skewed distribution



Summary of ScaLE properties

ScaLE has remarkable scaling properies for large data. BUT it does require

1. smoothness of the likelihood;
2. posterior contraction
3. to get the best scaling with dimension, require to find at least one point "close" to a mode of π.

Current implementation (in R!) is fairly slow and only suitable for fairly low dimensional parameter sets. But the scaling properties can be clearly seen.

Final comments

We have introduced two *principled subsampling* methods which exhibit iterative super-efficiency.

Note that with highly heterogenous data (eg all the information comes from a tiny fraction of the data) no method can be super-efficient.

ScaLE is statistically identical to the algorithm which would carry out no subsampling and fully the evaluate the target at each step. Zig-Zag is not statistically identical and can converge slower with subsampling.

Zig-Zag (and other PDMPS) are currently the more promising method for higher-dimensional problems.

Continuous-time algorithms involve many new implementational details and challenges. But these methods can often be more robust than their continuous-time competitors.

Final comment (continued)

Software:

https://github.com/mpoll/scale

RZigZag, see

https://diamweb.ewi.tudelft.nl/joris/pdmps.html

Current/future directions:

1. **ReScaLE**. While ScaLE uses a population approach (SMC) to realise the quasi-stationary distribution, ReScaLE is a single trajectory algorithm: rebirths come from the past trajectory rather than the remaining population of particles. Compared to ScaLE, ReScaLE is very fast, but has less robust convergence.
2. **Restore**. This is a pure non-reversible MCM algorithm invlolving rebirths together with local dynamics. http://arxiv.org/abs/1910.05037
3. Theoretical underpinning for all these methods!
4. Generic software using automatic differentiation.
5. Applications in infectious disease epidemiology