Constructing Efficient MCMC Methods Using Temporary Mapping and Caching

Radford M. Neal, University of Toronto
Dept. of Statistics and Dept. of Computer Science
`http://www.cs.utoronto.ca/~radford/`

Talk at Columbia University, Department of Statistics, 11 December 2006

Outline of This Talk

- Very brief introduction to Markov chain sampling.
- The idea of temporarily mapping to another space.
  - Existing methods that can be seen as doing this.
- The idea of caching previous probability calculations for later re-use.
  - Simple contexts where this is commonly done.
- Combining these two ideas in new ways.
  - Temporarily discretizing variables.
  - Handling distributions with ‘fast’ and ‘slow’ variables.
  - Mapping with a discretizing Markov chain, and its uses.
  - Adapting proposal distributions without really adapting.
  - Sampling using multi-point spaces.

Markov Chain Sampling

We wish to sample from some distribution for $x \in \mathcal{X}$ that has probability/density function $\pi(x)$.

Obtaining points drawn independently from π is too hard.

Instead we sample using a Markov chain with transition probabilities/densities (from x to x') denoted by $T(x'|x)$, for which π is an invariant distribution:

$$\int \pi(x)\, T(x'|x)\, dx \quad = \quad \pi(x')$$

If the chain is ergodic — ie, it will eventually visit all parts of the space from any starting point — it will converge to this (unique) invariant distribution.

We can obtain a sample of dependent points drawn approximately from π by simulating one or more chains for a suitably-long time, and then discarding the early parts of the chains. We can use this sample to estimate expectations or quantiles of functions of x.

Note that we can combine several transitions that leave π invariant by applying them in turn, or choosing one randomly at each step.

The Metropolis Algorithm

In the *Metropolis algorithm* a Markov chain transition from state x to state x' operates as follows:

1) A “candidate”, x^*, is proposed according to some probabilities $S(x^*|x)$, satisfying the symmetry condition that $S(x|x^*) = S(x^*|x)$.

2) This candidate, x^*, is accepted as the next state with probability

$$\min\left[1,\ \pi(x^*)/\pi(x)\right]$$

   If x^* is accepted, then $x' = x^*$. If x^* is instead rejected, then $x' = x$.

One can show that these transitions have π as an invariant distribution. Whether they are ergodic depends on the particular π and S.

A good choice of proposal distribution, S, is crucial to getting the chain to converge to π rapidly. A typical choice is for x^* to have a multivariate Gaussian distribution with mean x and covariance matrix σI. We need to choose a “proposal width”, σ, that is neither too big (almost all candidates rejected) nor too small (chain moves too slowly).

Transitions that Temporarily Map to Another Space

One way to define the transitions $T(x'|x)$ is via three other stochastic mappings, $\hat{T}$, $\bar{T}$, and $\check{T}$, as follows:

$$x \xrightarrow{\hat{T}} y \xrightarrow{\bar{T}} y' \xrightarrow{\check{T}} x'$$

Starting from x, we obtain a value in the temporary space by sampling from $\hat{T}(y|x)$. The target distribution for $y \in \mathcal{Y}$ has probability/density function $\rho(y)$. We require that

$$\int \pi(x)\,\hat{T}(y|x)\,dx \;=\; \rho(y)$$

$\bar{T}(y'|y)$ defines a transition on the temporary space that leaves ρ invariant:

$$\int \rho(y)\,\bar{T}(y'|y)\,dy \;=\; \rho(y')$$

Finally, $\check{T}$ gets us back to the original space. It must satisfy

$$\int \rho(y')\,\check{T}(x'|y')\,dy' \;=\; \pi(x')$$

The overall transition, $T(x'|x)$, leaves π invariant, and so can be used for Markov sampling of π.

Mapping to a Bigger Space: Auxiliary Variable Methods

The technique of introducing auxiliary variables can be seen in terms of a mapping from x to $y = (x, z)$, where z is a set of auxiliary variables. (Ie, $\mathcal{Y} = \mathcal{X} \otimes \mathcal{Z}$.)

We specify some conditional distribution for z given x, written $\rho(z|x)$, and then define $\rho(y) = \rho(x, z) = \pi(x)\rho(z|x)$. $\hat{T}$ leaves x unchanged, and samples a z to go with it, according to $\rho(z|x)$. $\check{T}$ just discards z, leaving x unchanged.

The point of the procedure is to be able to use a $\bar{T}$ that is more efficient than a transition for x defined directly.

Examples:

- Methods using Hamiltonian dynamics — z is a set of 'momentum' variables.
- Slice sampling — z is a vertical position under the plot of $\pi(x)$.
- Cluster methods —- eg, Swendsen-Wang algorithm for Ising models.
- Dirichlet process mixtures — z is a set of possible new clusters.

Mapping to a Smaller Space: Temporary Marginalization

Another technique is to temporarily marginalize away some variables.

Suppose $x = (u, y)$, so that $\mathcal{X} = \mathcal{U} \otimes \mathcal{Y}$. We can define $\hat{T}$ to just forget u, leaving y unchanged, so that $\rho(y)$ is the marginal distribution for y defined by π. $\check{T}$ will leave y unchanged and sample from $\pi(u|y)$.

The motivation: Updates for y alone may be more efficient, since it is of lower dimension than x. But these updates need to evaluate the marginal distribution for y, which may be slow, so we might not want to always work with y alone.

Example: Hidden Markov models, with u being the hidden state sequence.

The 'forward-backward' algorithm can evaluate the probability of an observed sequence summing over all possible u, but takes time proportional to $L \times N^2$ for a sequence of length L with N possible states. Updating each element of u in turn (with 'Gibbs sampling') takes only $L \times N$ time, but may move more slowly.

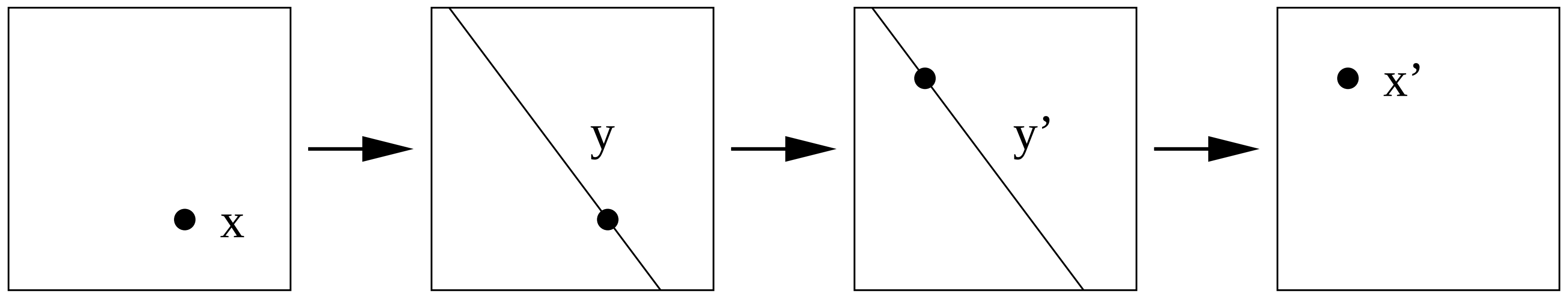
Since it's not obvious which approach is better, we might use both updates, mapping between $\mathcal{X}$ and $\mathcal{Y}$ as required.

Mapping to a Bigger But Effectively Smaller Space

We can also map to a bigger space, but then choose $\bar{T}$ to be **non-ergodic**, so that at any particular time, the space we map to is effectively smaller.

Example: 'Hit and run' methods. $\mathcal{X}$ is n-dimensional Euclidean space; $\mathcal{Y}$ is the space of lines in $\mathcal{X}$ together with a marked point on the line.

$\hat{T}$ randomly picks a line passing through x, with all directions equally likely, and makes x the marked point. $\check{T}$ sets x to the marked point on the line, then forgets the direction of the line. $\bar{T}$ moves the mark on the line, not changing the line itself.



Although $\bar{T}$ is non-ergodic, the full transition, T, in which the line direction is chosen randomly, may well be ergodic.

Caching Probability Evaluations for Re-use

Many transitions (eg, Metropolis updates) require evaluation of $\pi(x)$, up to some possibly-unknown normalization factor, for the current point and candidate points.

We can save ('cache') computed values of $\pi(x)$, and re-use them when needed.

Simple contexts where this can be beneficial:

- If we reject a Metropolis proposal, the next x will be the same as the current x. Saving the value of $\pi(x)$ avoids recomputing it.
- If we accept a Metropolis proposal, the next x will be the current candidate state, x^*. Saving $\pi(x^*)$ avoids computing $\pi(x)$ next time.
- Caching the values of $\pi(x^*)$ saves time if we propose the same x^* again.
- Caching values of $\pi(x)$ saves time if we backtrack, so that the x at a new time is the same as the x at some previous time.

The first two situations arise even for continuous spaces, and are handled efficiently by most MCMC implementations. The second two situations might seem relevant only when π is discrete. But mapping to another space can allow these sorts of gains from caching even when π is continuous.

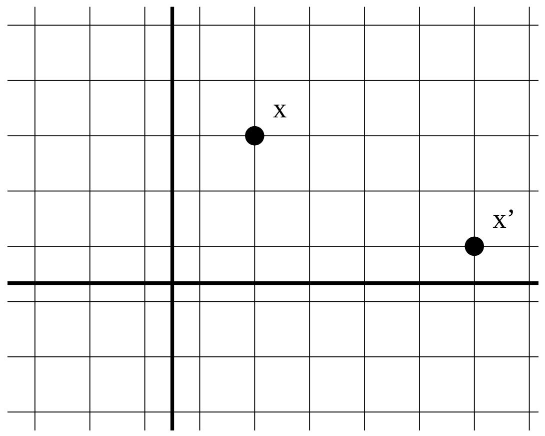
Temporarily Discretizing Variables

People sometimes replace continuous variables with discrete approximations (to save computation). We can do this with no approximation error using mappings.

Let $\mathcal{X}$ be n-dimensional Euclidean space. For some grid spacing, w, let $\mathcal{Y} = [0, w)^n \otimes Z^n$, where Z is the integers, so that $y = (s, z)$. $\hat{T}$ and $\check{T}$ are deterministic mappings, defined by $x = wz + s$. An example:

$$w = 1$$

$$\begin{aligned} x &= (1.5,\ 2.7) \\ s &= (0.5,\ 0.7), \quad z = (1,\ 2) \\ s' &= (0.5,\ 0.7), \quad z' = (5,\ 0) \\ x' &= (5.5,\ 0.7) \end{aligned}$$



Suppose we use a non-ergodic $\bar{T}$ that changes only z (the current grid point), while s (the position of the grid) fixed. With movement confined to a discrete grid, there will be non-zero probabilities of proposing the same candidate as before, and of backtracking to previous states, so caching can save computation.

We need to occasionally use some other update that can make small changes to x.

Distributions with ‘Fast’ and ‘Slow’ Variables

Consider a problem where $x = (u, v)$, with $u \in \mathcal{U}$ consists of one or more ‘slow’ variables, and $v \in \mathcal{V}$ consists of one or more ‘fast’ variables.

When we compute $\pi(u, v)$, we can save some intermediate results, and later compute $\pi(u, v')$ for any v' in much less time than would be needed to compute $\pi(u', v')$ for a new u'.

Example: Bayesian models of the cosmic microwave background radiation involve some parameters (u) of a complex simulation model as well as other parameters (v) that relate the output of the simulation to observations (see Lewis and Bridle). Finding the posterior density of $x = (u, v)$ for a new value of u requires running a new simulation. Finding the posterior density for a new value of v but an old value of u is much faster, if the simulation results for u were saved.

Example: Gaussian process classification models (and generalized linear mixed-effects models) involve latent variables with multivariate Gaussian distributions. Recomputing π after changing only the latent variables is faster than when the parameters of their covariance matrix changes.

Exploiting Fast Variables by Discretizing the Slow Variables

I suggested the following procedure, which has proved useful for the cosmic microwave background problem:

1) Randomly choose a direction, d, in the slow space, $\mathcal{U}$, and select some grid spacing, w.

2) Perform M Metropolis updates in which the proposal distribution sets $u^* = u \pm wd$, with the sign chosen randomly, and samples v^* from some symmetrical proposal distribution, $S(v^*|v)$.

3) Go back to step (1).

Because u^* will often be the same as a previous u^* or u, caching the information needed to quickly evaluate π with an old u will save time.

We can view this procedure as mapping to a space that includes the direction d, then discretizing in that direction. We could use a discretization of a higher dimensional subspace, but then the probability of re-using an old u might go down.

The transition, $\bar{T}$, defined by step (2) is non-ergodic, which here is crucial for obtaining the benefits of caching.

Mapping with a Discretizing Markov Chain

Rather than use a simple grid of points, we can discretize using a Markov chain designed to produce points that are more likely to be useful.

Let $R(x'|x)$ be the transition probabilities for such a chain, which leaves the distribution $\eta(x)$ invariant. Let $\tilde{R}(x|x') = R(x'|x)\eta(x)/\eta(x')$ be the reverse transition probabilities ($\tilde{R} = R$ for reversible chains).

We map from $\mathcal{X}$ to the space, $\mathcal{Y}$, of realizations of this discretizing Markov chain, of length K, with one time step of this chain being 'marked'. The current state, $x \in \mathcal{X}$, is mapped to a state in $\mathcal{Y}$ using a $\hat{T}$ that operates as follows:

1) Choose k uniformly from $0, \ldots, K-1$.

2) Simulate $K-1-k$ forward transitions, R, starting at $x_k = x$, producing states labeled $x_{k+1}, \ldots, x_{K-1}$.

3) Simulate k reverse transitions, $\tilde{R}$, starting at $x_k = x$, labeled $x_{k-1}, \ldots, x_0$.

4) Set the marked time step to k.

$\check{T}$ maps from $\mathcal{Y}$ to $\mathcal{X}$ by just taking the state at the marked time step.

Moving Along the Discretizing Chain

We can now define a non-ergodic transition, $\bar{T}$, on $\mathcal{Y}$ that only moves the mark, keeping the realization of the discretizing chain fixed. Effectively, $\bar{T}$ operates on the space $\{0, \ldots, K-1\}$ of positions on the realization of the discretizing chain.

These transitions must leave ρ invariant, where ρ has been implicitly defined as the distribution resulting from applying $\hat{T}$ to an x drawn from π.

If $\eta = \pi$, it's easy to show that $\bar{T}$ should leave the uniform distribution on $\{0, \ldots, K-1\}$ invariant.

More generally, $\bar{T}$ should leave invariant the distribution in which the mark is at time step k with probability proportional to $\pi(x_k) \,/\, \eta(x_k)$.

One possibility for $\bar{T}$ is a sequence of M Metropolis updates with proposal distribution that lets $k^* = k \pm 1$ with equal probabilities for the sign.

Note that we don't need to actually compute the whole discretizing chain at the beginning — we can extend it only as needed. We can then let K go to infinity.

Some Possible Applications (Mostly Untried)

- For a problem with fast and slow variables, we could apply a discretizing Markov chain on the slow space in which η contains only easily-computed factors of π (eg, just the prior). Proposals will be accepted based on the other factors of π. Note that in this application, $\bar{T}$ will propose both a new position of the mark and new values for the fast variables.

- If factors in the likelihood due to a subset of 'easy' observations can be computed faster than for other 'difficult' observation, we could let η be the posterior given only the easy observations. $\bar{T}$ would propose moving to fairly distant points on the discretizing chain, with acceptance determined by the factors of the likelihood due to the difficult observations.

- Even if all observations are similar, we could let η be the posterior using only a fraction f of the observations. The points on the discretizing chain will typically be spread out by a factor of $f^{-1/2}$ compared to π. If computing the likelihood takes time proportional to f^P, a speed-up may be expected if $D/2 < P$, where D is the dimensionality of the parameter space.

Embedded Hidden Markov Models

We can divide x into parts, as $x = (x_1, \ldots, x_\ell)$, and use a separate discretizing Markov chain for each x_i.

$\bar{T}$ will then operate on the product space of these discretizations, which contains K^ℓ points.

This is attractive when x is the space of hidden state sequences for a non-linear time series model with a large (perhaps continuous) state space. We can then make $\bar{T}$ sample a new sequence from the discretization *independently* of the current sequence, using a variation on the 'forward-backward' algorithm for this 'embedded Hidden Markov model', in time proportional to $K^2\ell$.

See the references for papers illustrating how this can improve sampling compared to simpler MCMC methods.

The Need to Adapt Proposal Distributions

Many MCMC methods have parameters that have to be tuned —eg, the proposal width for Metropolis updates. We seldom have completely-reliable *a priori* knowledge of the right values for these tuning parameters.

Typically, such parameters are tuned using preliminary MCMC runs, then kept fixed during the final runs. This works fairly well, but a better method would be nice.

Many people have investigated methods for adapting the tuning parameters *during* a run, based on statistics collected earlier in the run. Unfortunately, such adaptation undermines the Markov property needed to prove that the chain converges to π, and in general it doesn't.

Eventual convergence can be guaranteed if adaptation becomes slower and slower with time. But I wonder whether this has any real advantage over just fixing the parameters after a preliminary period of adaptation.

I propose another idea, which also allows different parameter values to (in effect) be used in different regions of the parameter space.

Using Caching to Effectively Adapt Without Really Adapting

Rather than adapt a tuning parameter (say σ), we can just cycle through m values of the parameter, $\sigma_1, \ldots, \sigma_m$, performing N_i updates using σ_i.

This way, we may sometimes use a good value of σ. Unfortunately, we also waste computation time using bad values of σ.

The idea: Use caching in conjunction with a mapping done with a discretizing Markov chain to drastically reduce the amount of computation time devoted to the bad values of σ, while still, in a mathematical sense, performing the pre-determined number of updates.

Since we don't adapt in a mathematical sense, convergence to π is still guaranteed.

Since we do adapt in a computational sense, we spend most of our time on updates that use a good value of σ.

A Sketch of How this ‘Short-Cut’ Method Works

I first worked out the details of how to take such a ‘short-cut’ for Metropolis in particular. I realize now that short-cuts can be taken for any MCMC method that has tuning parameters whose appropriateness can be observed.

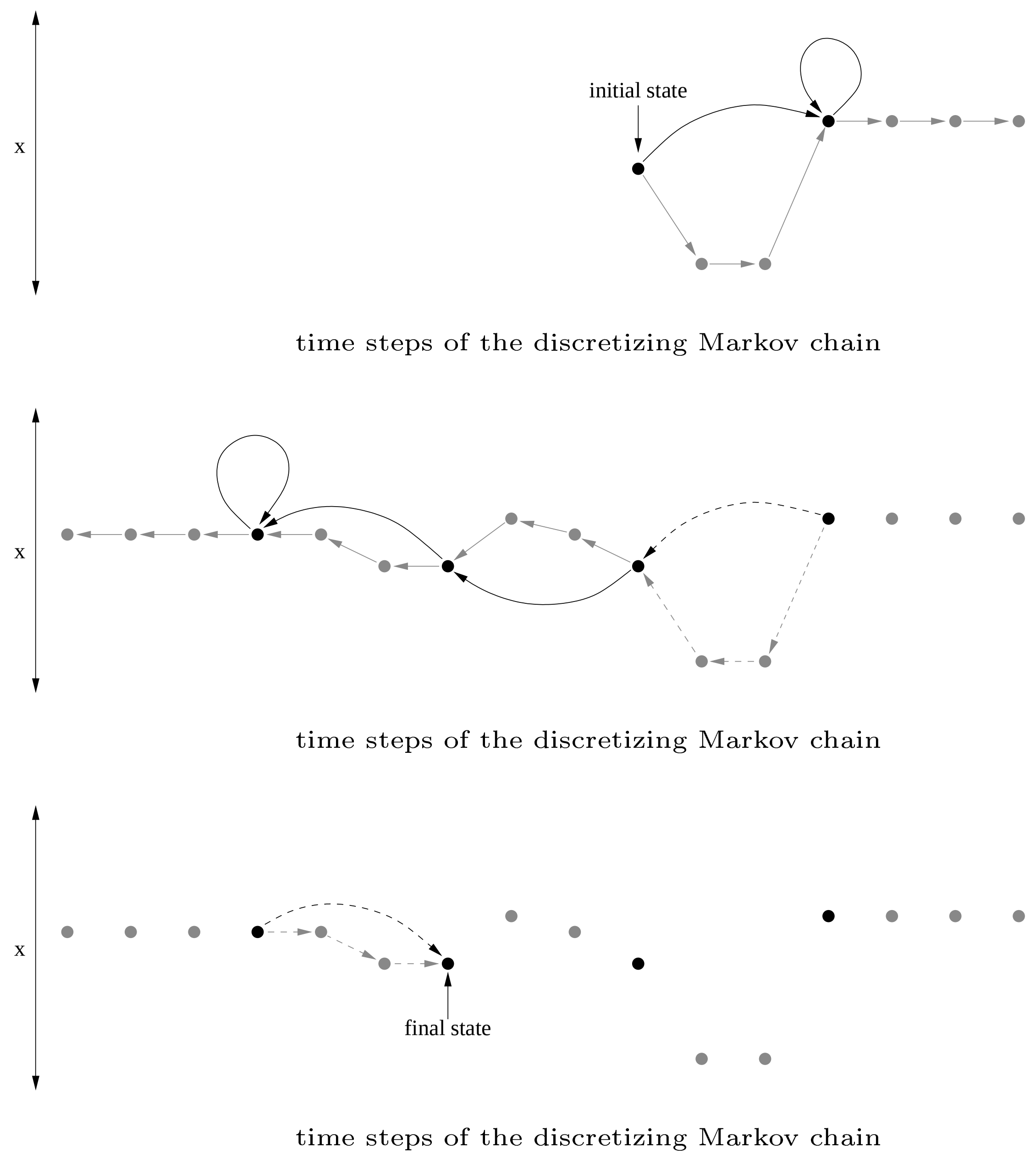
We map from $\mathcal{X}$ to the space of unbounded realizations of a discretizing Markov chain, R, that has π $(= \eta)$ as an invariant distribution, with one time step marked.

This setup is slightly modified so that we can look at short sequences (eg, ten long) of states from the discretizing Markov chain. From such a sequence, we can assess whether or not the tuning parameters being used (eg, σ_i) seem appropriate or not — for instance, by looking at the rejection rate in this short sequence.

The transitions $\bar{T}$ that move along the discretizing chain use a ‘direction’ flag that causes motion to continue in the same direction — *except* that the direction is reversed if data from the current short sequence indicates that the tuning parameters are bad.

After two such reversals, motion along the discretizing chain involves only states previously visited. If their values have been cached, no additional computation is required. So we spend little computation time on bad values of tuning parameters, but the mathematical properties needed for convergence still hold.

A Picture of Short-Cut Metropolis


initial state
x
time steps of the discretizing Markov chain
x
time steps of the discretizing Markov chain
final state
x
time steps of the discretizing Markov chain

Mapping to a Multi-Point Space

Let $\kappa(y_1, \ldots, y_K)$ be an exchangeable distribution (perhaps improper) on $\mathcal{Y} = \mathcal{X}^K$.

For example, if $K = 2$, κ could be uniform over all pairs (y_1, y_2) for which $||y_1 - y_2|| = d$, for some constant d.

We can define $\hat{T}$ as sampling $y_1, \ldots, y_K$ from its distribution under κ conditional on one of the y_k being equal to the current state, x. For the $K = 2$ example, we'd set one y_k to x and draw the other uniformly from the sphere of radius d around x.

It's easy to see that the distribution obtained from $\hat{T}$ is

$$\rho(y_1, \ldots, y_K) \;\;\propto\;\; \sum_{k=1}^{K} \pi(y_k)$$

Mapping back from $\mathcal{Y}$ to $\mathcal{X}$ is done using $\check{T}$, defined as

$$\check{T}(x' | y'_1, \ldots, y'_K) \;\;=\;\; \sum_{k=1}^{K} \delta(x', y'_k)\, \frac{\pi(y'_k)}{\pi(y'_1) + \cdots + \pi(y'_K)}$$

Uses of Multi-Point Mappings

Multi-point mappings seem like they might be useful in many ways.

For instance, you might hope that the pairs for the $K = 2$ example would both end up at places with high probability density, and that stepping in the direction they point would then be a good proposal.

Unfortunately, the simple form of $\hat{T}$ makes it clear that it won't be quite that easy. I'm presently thinking about whether some annealing scheme will allow this idea to work.

It does seem that multi-point mappings should allow for a MCMC form of the well-known Nelder-Mead simplex method for optimization without derivatives.

References

All my papers below are available from `http://www.cs.utoronto.ca/∼radford/`

R. M. Neal (1993) *Probabilistic Inference Using Markov Chain Monte Carlo Methods.*

General review of Markov chain Monte Carlo methods.

A. Lewis and S. Bridle (ongoing) Notes on the CosmoMC software, available at `http://cosmologist.info/notes/CosmoMC.pdf`.

Notes on MCMC for cosmological models of the cosmic microwave background.

R. M. Neal (2004) "Taking bigger Metropolis steps by dragging fast variables". Describes another way of handling fast and slow variables.

R. M. Neal (2003) "Markov chain sampling for non-linear state space models using embedded hidden Markov models".

Introduces the idea of embedded hidden Markov models. There's a conference paper (with Beal and Roweis) available as well.

R. M. Neal (2005) "The short-cut Metropolis method".

Introduces the short-cut method as applied to Metropolis updates, in a specialized way based on expressing Metropolis updates in terms of deterministic operations.