Distributed and Adaptive Darting Monte Carlo through Regenerations

Sungjin Ahn
Department of Computer Science
University of California, Irvine
Irvine, CA, USA
sungjia@ics.uci.edu

Yutian Chen
Department of Computer Science
University of California, Irvine
Irvine, CA, USA
yutianc@ics.uci.edu

Max Welling
Informatics Institute
University of Amsterdam
Amsterdam, The Netherlands
m.welling@uva.nl

Abstract

Darting Monte Carlo (DMC) is a MCMC procedure designed to effectively mix between multiple modes of a probability distribution. We propose an adaptive and distributed version of this method by using regenerations. This allows us to run multiple chains in parallel and adapt the shape of the jump regions as well as all other aspects of the Markov chain on the fly. We show that this significantly improves the performance of DMC because 1) a population of chains has a higher chance of finding the modes in the distribution, 2) jumping between modes becomes easier due to the adaptation of their shapes, 3) computation is much more efficient due to parallelization across multiple processors. While the curse of dimensionality is a challenge for both DMC and regeneration, we find that their combination ameliorates this issue slightly.

1 Introduction

Markov Chain Monte Carlo methods (MCMC) (Metropolis and Ulam, 1949) have revolutionized Bayesian statistics by providing a tractable approximation procedure for posterior inference (Gelfand and Smith, 1990). MCMC has also become an important tool for Bayesian approaches to machine learning (Andrieu et al., 2003). However, both in statistics as well as in machine learning, the challenges of today's large datasets remain a driving force for research in this area. To list a few research questions: 1) Can we effectively parallelize MCMC simulation across processors? 2) Can we efficiently mix between distant modes in the posterior? 3) Can we effectively adapt the Markov chain during execution?

Appearing in Proceedings of the 16^{th} International Conference on Artificial Intelligence and Statistics (AISTATS) 2013, Scottsdale, AZ, USA. Volume 31 of JMLR: W&CP 31.

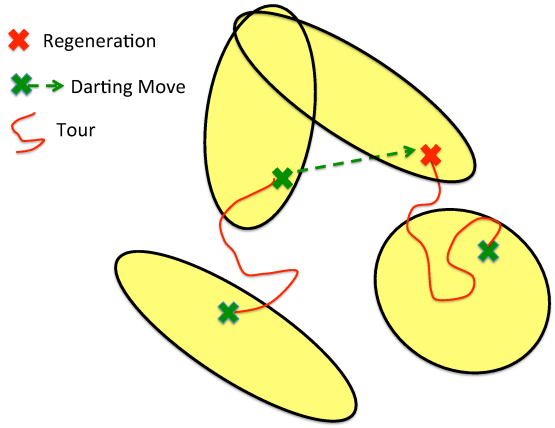


Figure 1: Illustration of how darting MCMC generates tours through regenerations.

In this paper we will touch upon all three of the above issues (parallelization, multi-modality, adaptation) by extending Darting Monte Carlo (Andricioaiei et al., 2001; Sminchisescu and M.Welling, 2007) through regeneration (Mykland et al., 1995; Gilks et al., 1998). Darting Monte Carlo (DMC) defines regions at locations of high posterior probability such that when entered (using a standard local MCMC move) a jump between regions will be attempted. This allows the algorithm to effectively move between distant modes. A drawback of the DMC algorithm is however that the jump regions need to be predefined and can not be adapted during a sampling run. This makes it impossible to *discover* new modes and include them on the fly. Ideally, one would have multiple Markov chains run *in parallel*, exploring different modes of the state space and regularly exchanging information in order to populate these modes in proportion to their volume. This idea is known as "population MCMC" (Warnes, 2001; Laskey and Myers, 2003; Braak, 2006) in the literature. However, these methods are not very flexible in the way they can adapt to previous samples. Moreover, they require a lot of communication between the parallel chains making them less

suitable for distributed simulation.

Regeneration is an alternative idea to parallelize MCMC simulation, with the additional bonus of being able to adapt all aspects of the Markov chain after each regeneration event. Enriching DMC with regenerations will allow the method to jump between regions, adapt the regions on the fly and simulate in parallel. Regeneration breaks a Markov chain into independent segments, or tours. At regeneration times one is allowed to adapt the details of the Markov chain (i.e. its transition kernel) based on available samples obtained so far (Gilks et al., 1998). We were inspired by ideas in Mykland et al. (1995) and Brockwell and Kadane (2005) to identify regeneration times. Regeneration has remained an elegant but somewhat impractical procedure that has not attracted a lot of attention (certainly not in the ML literature). Like DMC (or most mode jumping methods for that matter), it's main limitation is the curse of dimensionality. However, we argue that that regeneration combined with DMC is able to fine-tune its jump regions and ameliorate this problem to some degree, improving both DMC and regeneration MCMC.

2 Regeneration

Regeneration is an elegant method to break a Markov chain into smaller independent segments. (Gilks et al., 1998; Mykland et al., 1995; Brockwell and Kadane, 2005) The key idea is to split the kernel (i.e. proposal plus accept/reject step) into a mixture of two kernels as follows:

$$T(y|x) = S(x)Q(y) + (1 - S(x))R(y|x) \quad (1)$$

$$R(y|x) = \begin{cases} \dfrac{T(y|x) - S(x)Q(y)}{(1 - S(x))} & \text{IF } S(x) \in [0, 1) \\ 1 & \text{IF } S(x) = 1 \end{cases} \quad (2)$$

Here, $S(x)$ acts as a state dependent mixture coefficient between an independence kernel $Q(y)$ and the residual kernel $R(y|x)$. As long as we can find a factorization $S(x)Q(y) \leq T(y|x), \forall x, y$ then this construction is possible. If at any point we can interpret the sample as being generated by Q then this new sample is independent of the past and the chain has regenerated.

So far the construction is not practical because it is hard to sample from R. However, there is an elegant trick to decide on regeneration *retrospectively* which avoids sampling from R altogether. Introduce an auxiliary variable z_t that will decide which of the two kernels, Q or R, we choose. Thus, z_t follows a Bernoulli distribution with probability of success $S(x_{t-1})$. We now sample $(z_t, x_t)|x_{t-1}$ jointly. We may first integrate out z_t to sample $x_t|x_{t-1}$ and then retrospectively sample $z_t|(x_t, x_{t-1})$. Marginalizing out z_t will give us back the full mixture given by Eqn.1, i.e. we can simply propose a move according to T. Writing out the joint for (x_t, z_t, x_{t-1}) and conditioning on (x_t, x_{t-1}) one can show that,

$$z_t|(x_t, x_{t-1}) \sim \mathcal{B}\left(z; \frac{S(x_{t-1})Q(x_t)}{T(x_t|x_{t-1})}\right) \quad (3)$$

Remarkably, the procedure therefore simply uses the kernel T but decides whether a sample was a regeneration *retrospectively* by sampling from 3 and checking if $z_t = 1$. The tour that started at the previous regeneration and ends at x_{t-1} is independent of all other tours.

Detecting regenerations is very useful for a number of reasons. Firstly, the tours can be generated independently on different machines and combined later, facilitating parallel simulation (Brockwell and Kadane, 2005). But perhaps more importantly, all details of the entire procedure, including the local kernel (e.g. Hamiltonian Monte Carlo (HMC)) as well as the kernels T and Q can be adapted based on all the previous samples after a regeneration has occurred (Gilks et al., 1998).

The difficult part of designing a regenerative MCMC procedure is to find a valid factorization $S(x)Q(y) \leq T(y|x), \forall x, y$. However, this is possible when $T(y|x) = f(y)\alpha(x, y)$ with α the MH accept/reject step and $f(y)$ an independence sampler. Two methods were proposed for this in Mykland et al. (1995), which we describe in appendix A.

3 Darting Monte Carlo with Truncated DPMMs

To obtain high regeneration rate, it is of the utmost importance that the proposal distribution, f, (which we use as our independence sampler – see previous section) is a very tight fit to the target distribution π whenever we propose from the independence sampler. The strategy will be to tighten this fit at the regeneration times. In particular, we will fit a Dirichlet Process Mixture Model (DPMM) to the samples (or a suitable subset of the samples) that have been drawn so far. In particular, we used the "Accelerated Variational DPMM" algorithm of Kurihara et al. (2006) with a maximum number of clusters[1], K.

While we expect the DPMM to be a close fit in the high probability regions, the error may become much worse away from these regions (i.e. in the tails). Indeed, when the size of data set, N, is large (and under certain conditions), we may expect a mode to look normal only close to the local maximum. We therefore truncate (and re-normalize) the normal distributions obtained from the DPMM at α standard deviations from the mean and call these regions, $\mathcal{R} = \cup_i \mathcal{R}_i$, "jump regions". Because this "mixture of truncated normals" has zero probability outside $\mathcal{R}$ we do not

[1]Code available at `http://sites.google.com/site/kenichikurihara/academic-software/variational-dirichlet-process-gaussian-mixture-model`.

have to attempt sampling from f when the Markov chain is located outside of $\mathcal{R}$ (because the backward move has zero probability).

In order to encourage a high regeneration rate, we replace the mixture transition kernel in Sminchisescu and M.Welling (2007) by a cycle kernel, that is, at every iteration we first draw an intermediate sample, $\tilde{\theta}$, from a local sampler and then run a step of independence sampling only if $\tilde{\theta}$ is in $\mathcal{R}$. As shown in Mykland et al. (1995), the retrospective regeneration probability is not dependent on the local sampler but only on $\tilde{\theta}$.

Detailed balance and ergodicity can easily be proved. First, the proposed method uses a cycle kernel consisting of 2 kernels: the local HMC sampler and the truncated DPMM independence sampler (TDPMM), each of which is designed to satisfy detailed balance using a MH accept/reject step. Also, it is easy to see that this kernel is ergodic, since the first kernel (HMC) is ergodic and the second kernel (TDPMM) can move the chain to any location inside the jump regions.

The resulting "Regeneration Darting Monte Carlo" (RDMC) algorithm, which uses "method 1" in Appendix A is described in Algorithm 1.

Algorithm 1 Algorithm 1: Regeneration Darting Monte Carlo (RDMC)

Initialize θ_1
for $t = 1 : T$ **do**
  Sample $\tilde{\theta}$ according to local sampler (e.g. HMC).
  if $\tilde{\theta} \in \cup_i \mathcal{R}_i$ **then**
    Sample from TDPMM:
    1. Sample θ^* from: $f(\theta^*) \propto \sum_{i=1}^K \rho_i \, \mathcal{N}_i(\theta|\mu_i, \Sigma_i) \, \mathbb{I}[\theta \in \mathcal{R}_i]$
    2. Accept with probability $\alpha = \min\left[1, \frac{\pi(\theta^*) \sum_{i:\tilde{\theta}\in\mathcal{R}_i} \rho_i \mathcal{N}_i(\tilde{\theta}|\mu_i,\Sigma_i)}{\pi(\tilde{\theta}) \sum_{j:\theta^*\in\mathcal{R}_j} \rho_j \mathcal{N}_j(\theta^*|\mu_j,\Sigma_j)}\right]$.
    If accepted, use Eqn. 6 with $x = \tilde{\theta}, y = \theta^*$ to determine if the sample was a regeneration.
    if Regeneration has occurred **then**
      Adapt the MCMC kernel and discard θ^*.
      Apply rejection sampling using $f(\cdot)$ and Eqn. 7 to accept/reject in order to obtain θ_{t+1}.
    else
      Set $\theta_{t+1} = \theta^*$.
    end if
  end if
  If $\tilde{\theta} \notin \cup_i \mathcal{R}_i$ or θ^* is rejected, set $\theta_{t+1} = \tilde{\theta}$.
end for

(We note that $\sum_{i=1}^K \rho_i \; \mathcal{N}_i(\theta|\mu_i, \Sigma_i) \; \mathbb{I}[\theta \in \mathcal{R}_i] \propto \sum_{i=1}^K \zeta_i \; \mathcal{T}\mathcal{N}_i(\theta|\mu_i, \Sigma_i)$ with $\zeta_i = \rho_i \int_{\theta\in\mathcal{R}_i} \mathcal{N}_i(\theta)\mathrm{d}\theta$ so that jump proposals are indeed generated from a mixture of truncated Gaussians.) RDMC can indeed be interpreted as an improved "Darting Monte Carlo" (DMC) algorithm. Darting Monte Carlo was developed as an effective way for MCMC algorithms to jump between modes of a distribution. The initial paper (Andricioaiei et al., 2001) defined isotropic and uniform regions located close to the modes of the distribution. A local sampler, say HMC, is interrupted at regular intervals to check if the current location is inside one of the jump regions. If so, the DMC would proceed to propose a new value uniformly at random from within these regions followed by a standard Metropolis-Hastings (MH) accept/reject step. The procedure was generalized in Sminchisescu and M.Welling (2007) to handle overlapping regions of general shape. The independence sampler proposed in this paper (without regenerations and adaptation) improves on Sminchisescu and M.Welling (2007) by using truncated normal distributions inside the jump regions $\{\mathcal{R}_i\}$ instead of uniform probabilities, which provide a much better fit to the target distribution and is therefore expected to boost the acceptance rate. One can obtain the procedure from Sminchisescu and M.Welling (2007) by setting $\rho_i = 1$, $\mathcal{N}_i = \text{const.}$, $\forall i$ in which case $\sum_{i:\theta\in\mathcal{R}_i} 1 = n(\theta)$, the number of regions which include θ. As another special case one might consider no truncation, which would set $\zeta_i = \rho_i$.

We have experimented with an improvement to the basic form of RDMC. We exploits the property that any aspect of the MCMC procedure based on any information available at that time can be updated after a regeneration has occurred. This implies that in parallel to the Markov chains we can also run a number of mode searching optimization procedures, and incorporate newly found modes when updating the TDPMM. This flexibility to utilize different kinds of information in such a flexible manner seems unique to the regeneration procedure.

4 Experiments

In the following experiments, we study how the adaptation and parallelization in the proposed algorithm affect the convergence rate of the Markov chain. In particular, we provide experiment results on two models, Gaussian Mixture Model (GMM) and a localization problem of a Wireless Sensor Network (WSN). The convergence is diagnosed in two metrics: the multivariate potential scale reduction factor (MPSRF or R statistic) (Brooks and Gelman, 1998) and the relative error of the estimated mean (REM) of all dimensions. The R statistic is used to measure the convergence of multiple chains and its value approaches 1 when all chains converge to the stationary distribution. REM is a summary of the errors in approximating the expectation of variables across all dimensions computed as:

$$\text{REM}_t = \frac{\sum_{i=1}^d |\overline{\theta_i^t} - \theta_i^*|}{\sum_i |\theta_i^*|} \tag{4}$$

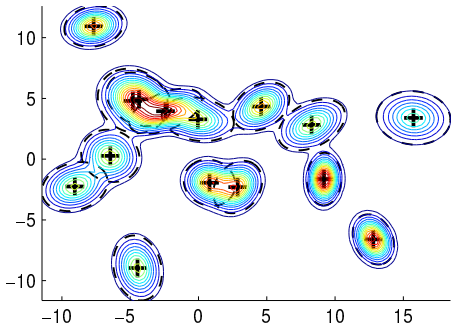
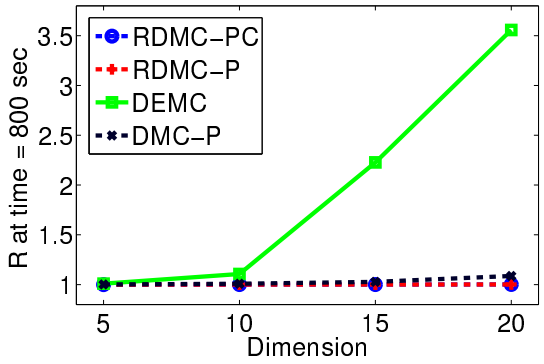
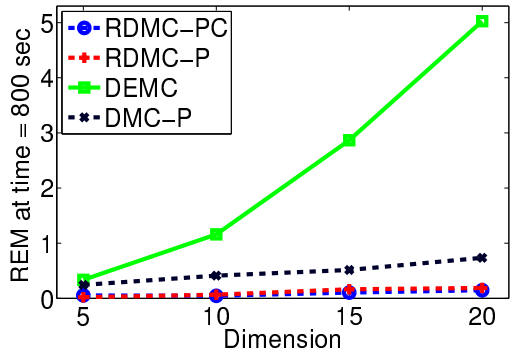


Figure 2: 2-D marginal of 15-component GMM



(a) R for increasing dimensions



(b) REM for increasing dimensions

Figure 3: R and REM of GMM for increasing dimension.

where $\overline{\theta_i^t}$ is the sampling average of i'th variable at time t, and θ_i^* is the mean w.r.t. the true distribution. For WSN where we cannot compute the true mean analytically, we simulate a long Markov chain using RDMC that covers all modes to provide a desirable precision.

We compare the following algorithms: 1) RDMC-PC(p) is our algorithm running p parallel chains and using combined tours when updating the DPMM. As a special case, RDMC-PC(1) runs a single chain on a single processor. To see the effect of combining tours among chains, we also consider 2) RDMC-P which is the same as RDMC-PC except that no communication is made between chains and thus DPMM is updated based on individual chains. 3) DMC-P(p) and 4) HMC-P(p) are respectively the original Darting algorithm (Sminchisescu and M.Welling, 2007) and Hamiltonian Monte Carlo running p chains in parallel. We also studied a population-based MCMC algorithm: 5) differential evolution MCMC (DEMC) (Braak, 2006), which allows distant mode jumps. DEMC(n) runs a population of size n on a single processor.

We briefly describe how the differential evolution MCMC works. Given a population of n samples, $\{\theta_i\}_{i=1}^n$, the proposed move for sample i is obtained by

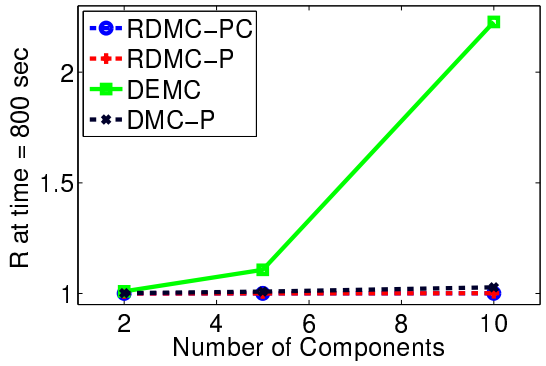
$$\theta_i^* = \theta_i + \gamma(\theta_j - \theta_k) + \varepsilon \tag{5}$$

where θ_j and θ_k (where, $i \neq j \neq k$) are chosen randomly from the population and the noise ε is subject to a Gaussian distribution $\mathcal{N}(0, b)$. Notice that $\theta_j - \theta_k$ determines the direction for θ_i to move and thus if θ_i and θ_k are in the same mode while θ_j in another, it will propose a jump for θ_i to a place near the mode of θ_j. When $\gamma = 0$, DEMC is equivalent to the random walk Metropolis sampler. Usually, b is set to a small number and $\gamma = 1$. Also notice that because of the high dependency among all samples, DEMC has to be executed sequentially on a single processor.

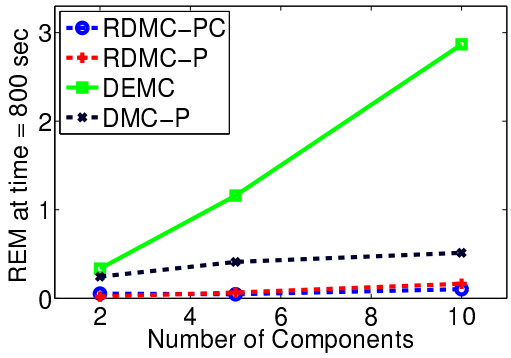
4.1 Gaussian Mixture Model

We first study how the algorithm is affected by varying the number of modes K and the dimension D. $K = [2, 5, 10]$ and $D = [5, 10, 15, 20]$ are considered for the comparison. When varying K or D, we fixed the other variable, $D = 10$ or $K = 5$, respectively. We randomly generate the Gaussian mixture models in such a way that the mean of each component is uniformly sampled from the d-dimension space while keeping the average distance among the components nearly constant for different K. Figure 2 shows a 2-d marginal distribution of a 15-component GMM.

We simulate 10 parallel chains, each on one processor, for every algorithm. The population size of DEMC, n, increases with K as $n = 20 + 10K$. For the HMC local kernel, we used 10 leapfrog steps and choose the stepsize to achieve an about 70% acceptance rate. For DMC and RDMC in order to rule out the possibility that the error is induced by sampling from different subset of modes, we perform a preliminary mode search until it finds all modes. For DMC, we first run gradient ascend algorithm with restart to find local modes, prune duplicate one, and then fit the jump regions, each centered at a mode with shape estimated by the Laplace method, as suggested in Sminchisescu and M.Welling (2007). For RDMC, we run a brief burn-in procedure, where we randomly initialize the samplers repeatedly, and run HMC to collect a few samples at every restart. The total set of samples are then used to



(a) R for increasing number of components



(b) REM for increasing number of components

Figure 4: R and REM of GMM for increasing number of components

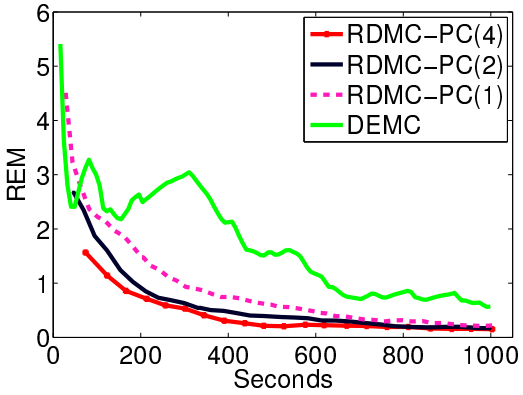


Figure 5: REM with mode search on the fly

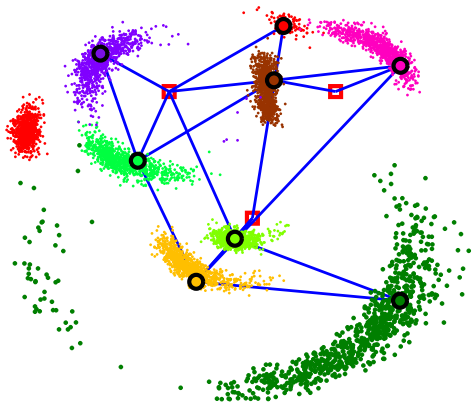


Figure 6: A network of 11 sensors with 3 known locations (red square) and 8 unknown (black circles). Point clouds show the marginal distribution of each sensor's location. The joint distribution is multi-modal and highly skewed.

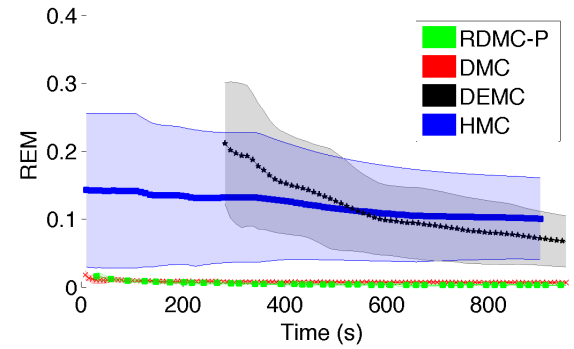
train an initial DPMM model. The time spent in this burn-in period is included in all time-related comparisons. We initialize the DMC and RDMC samples to be overdispersed so that it is possible to visit all the modes of the Gaussian mixture model.

Figure 3 and Figure 4 show the results after running the algorithm for 800 seconds. We can see that both of the RDMCs converge faster than DMC and DEMC in both R and REM. We updated the DPMM when a regeneration have occurred *and* the number of samples collected after the last adaptation is more than 2000. Although the effect of combining tours is reduced by the initial DPMM covering all the modes, by comparing RDMC-PC to RDMC-P we actually see that combining the tours improves the values slightly. Also, RDMC mitigates the curse of dimensionality problem that DMC is suffering from.

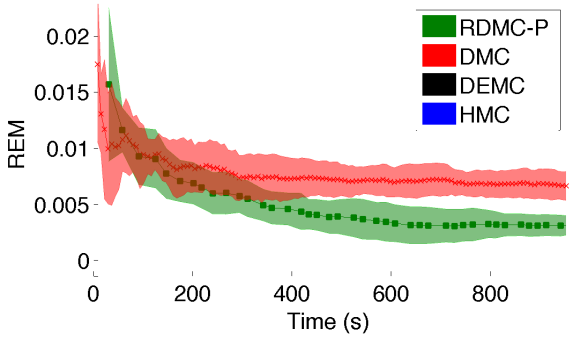
We also tested how the mode search on the fly (explained in the end of the Section 3) affects the convergence of the Markov chain. For this we tested RDMC-PC(1), RDMC-PC(2), RDMC-PC(4), and DEMC(n=100) on 8-component GMM. In this experiment, only one initial mode search is performed so that the maximum number of modes covered by the initial DPMM is equal to the number of parallel chains. For example, RDMC-PC(1) started with DPMM covering only one mode among eight modes. As shown in Fig.5, it however discovered all modes as the iteration goes on. Then, we increased the number of parallel chains up to 4 and the error decreased faster with this. This is because with the increasing number of parallel chains we can not only start with a DPMM with more modes, but also new modes can be discovered faster with multiple mode search.

4.2 Sensor Network Localization

In this section we illustrate the advantage of our adaptive algorithm in a simulated problem of sensor network localization. Following the experiment setting in Ihler et al. (2005), assume N sensors are scattered in a planar region with two-dimensional locations denoted by $\{\mathbf{x}_t\}_{t=1}^N$. The distance between a pair of sensors $(\mathbf{x}_t, \mathbf{x}_u)$ is observed with a probability $P_o(\mathbf{x}_t, \mathbf{x}_u) = \exp(-.5\|\mathbf{x}_t - \mathbf{x}_u\|^2/R^2)$, and the observed distance is corrupted by Gaussian noise:



(a)



(b)

Figure 7: Relative error of the estimated posterior mean of sensor locations. The mean and standard deviation are computed from 10 Markov chains. The bottom figure is a zoom-in view of the top figure. The first point on each figure indicates the time for the burn-in period.

$d_{tu} = \|\mathbf{x}_t - \mathbf{x}_u\| + \nu_{tu}$, $\nu_{tu} \sim \mathcal{N}(0, \sigma_\nu^2)$. Given a set of observations $\{d_{tu}\}$ and a prior distribution for x_t, a uniform distribution in this paper, a typical task is to infer the posterior joint distribution of all the sensor locations. We choose $N = 8$, $R/L = .3$, $\sigma_\nu/L = .02$ and add three additional sensors with known locations to avoid the ambiguities of translation, rotation, and negation (mirror symmetry). The locations of the 8 sensors form a multi-modal distribution of 16 dimensions, with their marginal distribution displayed in Figure 6.

We use the same criterion as in the previous section to optimize the HMC local sampler and do the mode search for DMC and RDMC. For the population MCMC algorithm, DEMC, the population size is selected as 100 to balance the efficiency and jumping acceptance rate. We set the standard deviation of the Gaussian noise as 5×10^{-5} to achieve an acceptance rate of about 40% for the random walk proposal. 10 Markov chains are simulated for every algorithm.

We compare the errors of estimating the posterior mean of the sensor locations as a function of time in Figure 7 and also show the corresponding R statistic in Figure 8.

As there are separated local modes in the posterior distribution (e.g. the two red clusters in Figure 6) pure local samplers such as HMC cannot visit all modes, resulting in a large bias. Consistently the R statistic stays at a large value indicating that multiple HMC chains do not mix.

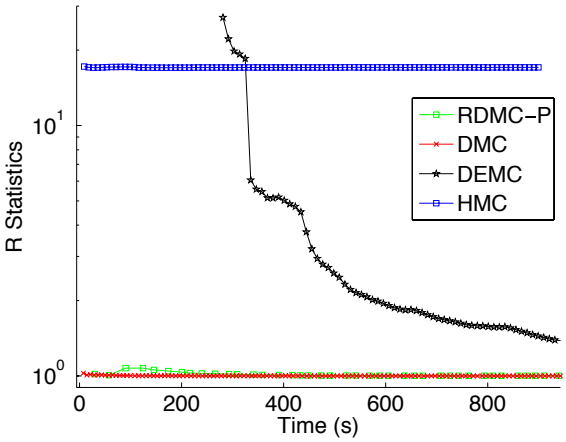


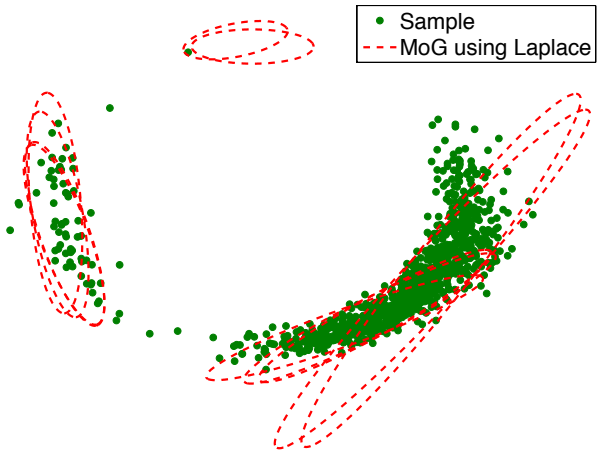
Figure 8: R statistic of 10 Markov chains as a function of time.

For the population MCMC method, DEMC, we use 100 particles for each chain to encourage mode jumping, which in return slows down the algorithm considerably. In order to make sure the samplers could still move locally when the jump proposal is rejected, we improve the algorithm by decomposing the transition kernel of DEMC into two consecutive steps, local random walk and jump proposal, each followed by a Metropolis-Hastings step. However, even though we do observe the samplers jump from one mode to another occasionally, the acceptance rate is still very low ($\sim 10^{-3}$). The slow convergence leads to large variance in the estimated mean and a slow decay of the R value.

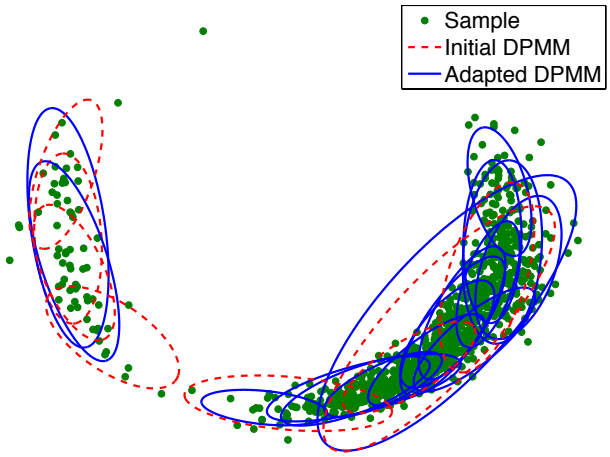
In contrast with the two methods above, all the darting based algorithms show fast convergence in both the estimated error and the R statistic. Moreover, the regeneration algorithm converges faster than the original darting Monte Carlo algorithm. As the proposal distribution in RDMC is adapted with more samples, the difference of these two curves become more significant.

We study the proposal distributions qualitatively in Figure 9. This figure shows a 2-D projection of the jumping regions at the 2 dimensions corresponding to the 5th sensor. Apparently, the mixture of Gaussian model trained by DPMM is a tighter fit to the underlying distribution. Moreover, DPMM keeps adapted and improved as more samples are collected. A tighter proposal distribution provides both higher jumping acceptance rate and higher regeneration rate, which eventually leads to an improved convergence rate of the Markov chain.

The same conclusion can be made by looking at the regeneration rate in Figure 10. The regeneration rate measures



(a) Mixture of Gaussian fitted by Laplace after burn-in



(b) Mixture of Gaussian fitted by DPMM after burn-in (red) and after 1000 seconds (blue).

Figure 9: Marginal distribution of the 5th sensor (green points) and the 2-D projection of the one standard deviation ellipses of the mixture of Gaussian model on the sensor's location.

the frequency of an independent sample generated from the Markov chain. A higher rate indicates a faster convergence rate. We find that the regeneration rate of RDMC increases as the mixture model keeps adapted to the true distribution.

In practical problems with multiple modes, we are not able to find all the modes in a burn-in period. Figure 11 shows the case when we cannot find all the modes. Each chain runs a single mode search, and none of them could find all the mode. By communicating and exchanging information among chains, RDMC-PC is able to find more modes and thereby reduce the estimation error.

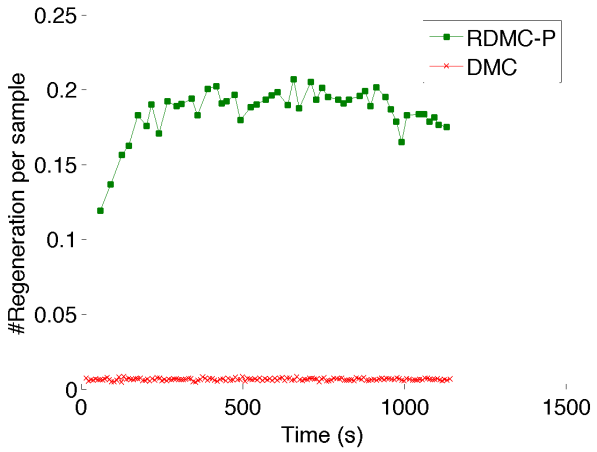


Figure 10: The average and standard deviation of the regeneration rate for DMC and RDMC.

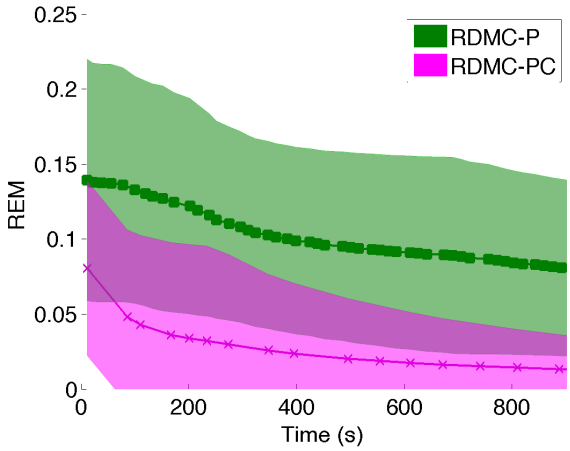


Figure 11: Relative error of the estimated posterior mean of sensor locations. Each chain runs one mode search.

5 Conclusion

In the machine learning community, regeneration has not made its appearance so far to the best of our knowledge[2] Yet, regenerations provide a elegant method to parallelize and adapt MCMC procedures. We found that it was particularly powerful in combination with DMC because the jump regions can now be adapted to the shape of the distribution, causing both more jumps and more regenerations.

A challenge for both DMC and RDMC is the curse of dimensionality. Due to the fact that RDMC was able to provide tighter fits to the shape of the mode this problem was slightly ameliorated relative to DMC. We find in extended

[2]One paper is similar in spirit to our procedure, called "Variational MCMC" (de Freitas et al., 2001) where a variational approximation acts as the proposal for an independence sampler. Regenerations are mentioned as a possible way to adapt and improve this proposal.

experiments that RDMC can run effectively up to 50 dimensions, which is an order of magnitude more than the empirical findings of Gilks et al. (1998); Mykland et al. (1995); Brockwell and Kadane (2005). Future research will be directed towards further improving this issue.

A Regenerations from an Independence Sampler

Below we provide details of two regeneration methods based on the independence sampler (Mykland et al., 1995). Define $w(\cdot) = \pi(\cdot)/f(\cdot)$, where π is the target distribution.

Method 1: When $y \sim f$ is accepted according to α, the probability of a regeneration is given by,

$$P_{\text{reg}} = \begin{cases} 1 & \text{IF } w(x) \geq c, w(y) \leq c \\ & \text{OR } w(x) \leq c, w(y) \geq c \\ \frac{1}{c}\max\left[w(x), w(y)\right] & \text{IF } w(x) < c, w(y) < c \\ c\max\left[\frac{1}{w(x)}, \frac{1}{w(y)}\right] & \text{IF} w(x) > c, w(y) > c \end{cases} \tag{6}$$

where c is an arbitrary constant which should set to maximize the probability of regeneration. A reasonable choice is $c = \mathbb{E}_\pi[w]$ which can be approximated from samples and adapted after each regeneration.

If we adapt the transition kernel after a regeneration has occurred, then the last sample y obtained from the old kernel should be discarded, and a new sample from the independence sampler $Q(y)$ should be drawn. We can obtain y by rejection sampling where we repeatedly propose y from $f(y)$ until it is accepted with probability (Gilks et al., 1998)

$$P_{\text{new}} = \min\left[1, \frac{w(y)}{c}\right] \tag{7}$$

Method 2: Define an "envelope function" g from the proposal f as follows: $g(y) = mf(y)$, with $m > 1$ a constant such that close to the modes we have $g(y) > \pi(y)$. We call $\mathcal{C}$ the set where $g(y) \geq \pi(y)$. Since f is a proper density and g is just equal to f up to a multiplicative constant m, samples are repeatedly proposed from f until one sample gets accepted according to,

$$P_{\text{accept}} = \min\left[1, \frac{w(y)}{m}\right] \tag{8}$$

This proposal thus samples from a distribution proportional to $q(y) \propto \min[\pi(y), g(y)]$ because if $g(y) \geq \pi(y)$ then it samples correctly from π using standard rejection sampling, but if $g(y) < \pi(y)$ it incorrectly accepts the sample drawn from g. Next, we need to accept or reject this proposed sample using a standard MH step (Tierney, 1994),

$$P_{\text{accept}} = \begin{cases} 1 & \text{IF } \ x \in \mathcal{C} \\ \frac{m}{w(x)} & \text{IF } \ x \notin \mathcal{C}, y \in \mathcal{C} \\ \min\left[1, \frac{w(y)}{w(x)}\right] & \text{IF } \ x \notin \mathcal{C}, y \notin \mathcal{C} \end{cases} \tag{9}$$

If accepted, we then determine if we have regenerated (using q instead of f in method 1, Eqn. 6 and setting $c = 1$), leading to

$$P_{\text{reg}} = \begin{cases} 1 & \text{IF } \ x \in \mathcal{C} \ \text{ OR } \ y \in \mathcal{C} \\ m\max[\frac{1}{w(x)}, \frac{1}{w(y)}] & \text{otherwise}^3 \end{cases} \tag{10}$$

Note that all accepted samples that fall in the region $\mathcal{C}$ are regenerations.

Similar to method 1, after adaptation we should draw a new sample of y from $Q(y)$. This implies sampling from $q(y) = \min\left[\pi(y), g(y)\right]$ (using the rejection sampling procedure described above) and then simply accepting that sample (because $P_{\text{new}} = \min\left[\frac{\pi(y)}{\min[\pi(y), g(y)]}, 1\right] = 1$ in this case)

References

N. Metropolis and S. Ulam. The monte carlo method. *Journal of the American Statistical Association*, 44(247): 335–341, 1949.

A.E. Gelfand and A.F.M. Smith. Sampling-based approaches to calculating marginal densities. *J. American Statistical Association*, 85:398–409, 1990.

C. Andrieu, N. de Freitas, A. Doucet, and M.I. Jordan. An introduction to mcmc for machine learning. *Machine Learning*, 50:5–43, 2003.

I. Andricioaiei, J. Straub, and A. Voter. Smart darting montecarlo. 114(16), 2001.

C. Sminchisescu and M.Welling. Generalized darting monte carlo. In *Eleventh International Conference on Artificial Intelligence and Statistics (AISTATS2007)*, 2007. online proceedings.

P. Mykland, L. Tierney, and B. Yu. Regeneration in markov chain samplers. *Journal of the American Statistical Association*, 90(429):233–241, 1995.

W.R. Gilks, G.O. Roberts, and S.K. Sahu. Adaptive markov chain monte carlo through regeneration. *J. Amer. Statist. Assoc.*, 93:1045–1054, 1998.

G.R. Warnes. The normal kernel coupler: An adaptive markov chain monte carlo method for efficiently sampling from multi-modal distributions. Technical report, University of Washington Department of Statistics, 2001. Technical Report no. 395.

[3] There is a typo in computing r_A in Mykland et al. (1995): "min" should be replaced by "max".

K. B. Laskey and J. W. Myers. Population markov chain monte carlo. *Machine Learning*, 50:175–196, 2003.

Cajo J. F. Ter Braak. A markov chain monte carlo version of the genetic algorithm differential evolution: easy bayesian computing for real parameter spaces. *Statistical Computing*, 2006.

A.E. Brockwell and J.B. Kadane. Identification of regeneration times in mcmc simulation, with application to adaptive schemes. *Journal of Computational and Graphical Statistics*, 14(2):436–458, 2005.

K. Kurihara, M. Welling, and N. Vlassis. Accelerated variational Dirichlet process mixtures. In *Advances of Neural Information Processing Systems – NIPS*, volume 19, 2006.

S.P. Brooks and A. Gelman. General methods for monitoring convergence of iterative simulations. *Journal of computational and graphical statistics*, 7(4):434–455, 1998.

A.T. Ihler, J.W. Fisher III, R.L. Moses, and A.S. Willsky. Nonparametric belief propagation for self-localization of sensor networks. *Selected Areas in Communications, IEEE Journal on*, 23(4):809–819, 2005.

Nando de Freitas, Pedro A. d. F. R. Hojen-Sorensen, and Stuart J. Russell. Variational mcmc. In *UAI '01: Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence*, pages 120–127, 2001.

L. Tierney. Markov chains for exploring posterior distributions. *Annals of Statistics*, 22(4):1701–1728, 1994.