Leveraging Communication for Efficient Sampling

by

Sanyam Kapoor

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science

Computer Science Department
New York University
May 2019

Advisor: Professor Joan Bruna

Second Reader: Professor Kyunghyun Cho

Acknowledgements

I would like to wholeheartedly thank Professor Joan Bruna for advising me on this thesis and getting me started with research. Thank you for letting me pursue half-baked ideas and nudging me in the right directions all the time. I've been interested in the problem of exploration for a while and he has been highly influential in helping me approach the right formulations. I would like to thank Professor Kyunghyun Cho for agreeing to be the second reader.

I would like to thank Cinjon Resnick, Roberta Raileanu and Ilya Kostrikov for being great discussion partners and getting me started in Reinforcement Learning. I've had a great deal to learn from them.

Finally, none of this would have been remotely possible without the everlasting support and pep talks of my parents, Abha Kapoor and Sunil Kapoor.

Abstract

Machine Learning has shown promising success tasks like classification, regression and more recently generation. However, long-term planning still remains a challenge for real-world deployment and one of the key components of long-term planning is exploration. In this work, we discuss how *communication* can be leveraged to improve space exploration. We study this problem from the perspective of sampling from un-normalized density functions. *Hamiltonian Monte Carlo* (HMC) finds it improbable to sample from highly separated multi-modal distributions and parallel chains can be wasteful by the nature of Markov chain sampling. We see how replica exchange induces a weak for of communication. This is contrasted with a particle based approach called the *Stein Variational Gradient Descent* (SVGD) which induces a stronger form of communication via kernel evaluations. The quality of samples from both HMC and SVGD are evaluated with *Maximum Mean Discrepancy*. We finally propose Graph Neural Networks with stronger inductive biases to amortize the dynamics of SVGD for fast generation of representative samples.

Contents

List of Figures

Chapter 1

Introduction

Machine learning (ML) has had tremendous impact in recent years. Particularly, reinforcement learning has demonstrated great success by solving long-term planning games like Go (Silver et al. 2017), Starcraft II (Vinyals et al. 2019) and Dota 2 (OpenAI 2018). These works are solid proofs-of-concept because they present an approach without any structured rule available to the learning algorithm. They present experimental evidence on how combinatorial generalization is imperative for future AI systems (Battaglia et al. 2018) and the presence (or lack thereof) of inductive biases in our algorithms have led us so far. A consequence of weak inductive biases is however the need for massive compute. AlphaStar (Vinyals et al. 2019) reports an average of 200 years worth of gameplay per agent. Clearly, this is unlike humans and we need to appeal to all elements of intelligence (Sukhbaatar 2018) - memory, communication and intrinsic motivation.

In this work, we investigate how *communication* can affect the efficiency of learning systems. We first discuss the motivation for our work and defer the exact notion of communication relevant to our study.

1.1 Motivation

We take a foundational approach to studying how communication affects the performance of a system and discuss an abstract analogy first. We consider the problem of sampling from a probability density function. The objective of any sampling algorithm is to provide an efficient means to compute a statistic of interest under a given target distribution. For instance, to compute the expected value of a random function f under a distribution p over the reals $\mathbb{R}$, we need the estimate of $\bar{f}$ defined as

$$\bar{f} = \mathbb{E}_p \left[f(X) \right] \tag{1.1}$$

By definition, this is simply taking the intergral over the full real space. A naive approximation is done via Monte Carlo samples $x_i \sim p(x)$ combined with the weak law of large numbers and leads to

$$\begin{aligned} \bar{f} &= \int_{\mathcal{X}} f(x)p(x)dx \\ &\approx \frac{1}{N} \sum_{i=1}^{N} f(x_i) \end{aligned} \tag{1.2}$$

However, low density regions contribute very little to this expectation. We instead require samples from a high volume region known as the *typical set* (Betancourt 2017) and need a more *informed* routine to sample the most important parts of the space.

Analogously, we need an *agent* that can traverse and find the high density regions within a prescribed computational budget. This task is more directly motivated when we view the agent's policies as a Gibbs distribution induced by the action-value function (Sutton and Barto 2018) of the following form (Haarnoja et al. 2017)

$$\pi(\mathbf{a}_t|\mathbf{s_t}) \propto \exp\left\{ \frac{1}{\alpha} Q(\mathbf{s}_t, \mathbf{a}_t) \right\} \tag{1.3}$$

In principle, quite naturally, this formulation encourages the agent to sample actions that can lead to higher *action value*s in expectation. To build good estimates of such functions, one of the imperatives to efficient reinforcement learning systems is efficient exploration. While we do not explicitly formulate reinforcement learning problems in this work, we hypothesize that it has much to gain from sampling literature.

1.2 Connections to sampling

The rest of the work is organized around investigating the dynamics of particles (aka agents) in density spaces and see how well they can explore. We consider toy density problems like mixture of highly separated Gaussians and synthetic distributions like the ring and mixture of rings. We keep the following assumptions in our experiments

- Density spaces are continuous.
- We can query the oracle for unnormalized density function $\tilde{p}(x)$.
- We can compute derivatives of these density functions $\nabla\tilde{p}(x)$.

These assumptions are standard to many formulations in machine learning literature. As a concrete example, Bayesian inference (Murphy 2012) requires us to compute the posterior distribution over the parameter set θ given data $\mathcal{D}$ and further sample for the posterior predictive distribution over a new sample x'

$$p(\theta|\mathcal{D}) = \frac{p(\mathcal{D}|\theta)p(\theta)}{p(\mathcal{D})} \tag{1.4}$$

$$p(y'|x', \mathcal{D}) = \int p(y'|x', \theta)p(\theta|\mathcal{D})d\theta \tag{1.5}$$

$p(\mathcal{D})$ is intractable for almost all problems of practical interest. Hence, we consider the unnormalized posterior $\tilde{p}(\theta|\mathcal{D}) = p(\mathcal{D}|\theta)p(\theta)$ and our objective is to *explore* this posterior well to build a good estimate for the predictive distribution from posterior samples.

In the rest of the work, we revisit Hamiltonian Monte Carlo (HMC) sampling and see how parallel tempering induces a primitive form of communication. We also understand the limitations of HMC and see how Stein Variational Gradient Descent (SVGD) imposes a stronger form of communication via kernel evaluations. We finally propose amortization of this communication into graph neural networks for efficient sampling.

Chapter 2

Hamiltonian Monte Carlo

In this chapter, we revisit Hamiltonian Monte Carlo (Duane et al. 1987), one of the most popular techniques from the family of Markov Chain Monte Carlo algorithms (Metropolis et al. 1953) to sample from unnormalized density functions.

2.1 Background

The key inspiration behind HMC comes from the physical notion of *Hamiltonian* H of a system. We augment the space of interest x with momentum variables p of equal dimension colloqiually known as the phase space $z = (x, p)$ and simulate the following differential equations

$$\frac{dz}{dt} = \begin{bmatrix} \frac{dx}{dt} \\ \frac{dp}{dt} \end{bmatrix} = \begin{bmatrix} \frac{\partial H}{\partial p} \\ -\frac{\partial H}{\partial x} \end{bmatrix} \tag{2.1}$$

where the *Hamiltonian* H is a sum of *potential* energy $U(x) = -\log \tilde{p}(x)$ and *kinetic* energy $K(p) = p^T M^{-1} p/2$. Note the use of an unnormalized density function. Typically, the mass matrix M is considered to be identity I. Application of this differential equation in practice

requires discretization, however the basic Euler's method leads to diverging trajectories. Instead, we use a modified routine known as the leapfrog method (Neal et al. 2011) with discretization steps of ϵ

$$p(t + \epsilon/2) = p(t) - (\epsilon/2)\frac{\partial U(x(t))}{\partial x} \tag{2.2}$$

$$x(t + \epsilon) = x(t) + \epsilon p(t + \epsilon/2) \tag{2.3}$$

$$p(t + \epsilon) = p(t + \epsilon/2) - (\epsilon/2)\frac{\partial U(x(t + \epsilon))}{\partial x} \tag{2.4}$$

We finally need a Metropolis-Hastings correction given by

$$A(x \to x') = \min\ (1, \exp\{-H(z') + H(z)\}) \tag{2.5}$$

where z and z' are the augmented phase space variables with corresponding position as x and x' respectively.

The exact HMC step with the leapfrog integrator and the MH acceptance ratio is given in Algorithm 2.1.

The HMC proposal satisfies the detailed balance and hence the stationary distribution is the one induced by the density function $\tilde{p}$. For detailed proofs refer Neal et al. 2011; Murphy 2012. These steps are repeated to create a Markov chain. In principle, HMC builds chains where the particle stays on iso-probability contours of the phase space. At each step of the HMC step as seen in Algorithm 2.1, we resample momentum and effectively jump around these contours. Betancourt 2017 discusses the geometry of Hamiltonian Monte Carlo in greater detail. It is this same behavior that induces some pathologies in HMC.

- Due to the volume preserving transforms (Neal et al. 2011), HMC mixes poorly across energy levels

Algorithm 2.1 Hamiltonian Monte Carlo Step

Input: An unnormalized probability density function $\tilde{p}$, leapfrog integrator steps τ, time discretization ϵ

procedure HMC-STEP(x)
  $x_0 \leftarrow x$
  $p_0 \sim \mathcal{N}(0, I)$ ▷ Initialize momentum
  $H_0 \leftarrow -\log \tilde{p}(x_0) + p_0^T p_0/2$ ▷ Initial Hamiltonian
  for t `in` 1 `to` τ **do** ▷ Leapfrog integrator
    $p_{t_{1/2}} \leftarrow p_{t-1} + \frac{\epsilon}{2} \nabla_x \log \tilde{p}(x_{t-1})$
    $x_t \leftarrow x_{t-1} + \epsilon p_{t_{1/2}}$
    $p_t \leftarrow p_{t_{1/2}} + \frac{\epsilon}{2} \nabla_x \log \tilde{p}(x_t)$
  $H_\tau \leftarrow -\log \tilde{p}(x_\tau) + p_\tau^T p_\tau/2$ ▷ Final Hamiltonian
  $a \sim \mathcal{U}(0, 1)$
  if $a < A(x_0 \rightarrow x_\tau)$ **then** ▷ Acceptance step
    return x_τ
  else
    return x_0

- HMC usually fails at sampling highly separated multi-modal distributions (like a mixture of Gaussian) as the probability of sampling large momentum to jump the low density regions is very low
- HMC struggles with ill-conditioned landscapes (Girolami and Calderhead 2011)
- HMC deals poorly with rapidly changing gradients (Sohl-Dickstein, Mudigonda, and DeWeese 2014)

HMC requires careful tuning of τ and ϵ and this involves multiple initial chains to monitor trace plots. We visit a few toy distributions and see how HMC behaves in experiments later (§2.3).

2.2 Parallel Tempering

To help HMC cover larger parts of the space, we can sample independent and parallel chains. Under an over-dispersed initialization of such chains, they will likely traverse different parts of the space especially when the distribution is multi-modal. Each chain can in principle collapse into one of the many modes especially with multi-modal landscapes and find it hard to jump out owing to ill-conditioned geometry. Naturally, stochasticity can also cause multiple chains to collapse into the same mode as well. Clearly, this lack of information sharing across chains is wasteful.

A very primitive way to share information instead is if we build multiple landscapes and exchange replicas of particles across chains, we may be able to jump over low density regions where a chain may not particularly mix fast enough. Multiple landscapes are generated by varying temperatures of the system. This replica exchange leads to an idea known as *parallel tempering* (Swendsen and J.-S. Wang 1986). Here again, we use the augmented phase space from HMC and consider a physical property temperature. We characterize the Gibbs distribution (unnormalized) of the system as

$$\tilde{p}_T(x) = \exp\left\{-\frac{1}{T}\log\tilde{p}(x)\right\} \tag{2.6}$$

Using $T = 1$ we can recover the original augmented distribution $\tilde{p}(z)$. In the limit $T \to \infty$, $p_T(z)$ approaches a uniform distribution. Intuitively, increasing the temperature makes the phase space landscape becomes nicer and easier to traverse for a particle using gradient information as it tends to flatten the sharp curvatures. During a *parallel tempering* run with HMC, we "exchange replicas" to virtually jump across phase spaces. A more exhaustive analysis from the perspective of physical applications is presented in Earl and Deem 2005. Replica exchange is implemented as an additional step in the chain simulation by computing the probability of the exchange between two systems at temperatures T and T' by the ratio

Algorithm 2.2 Parallel Tempering Step

Input: An unnormalized probability density function $\tilde{p}$, temperatures for N parallel chains $\{T_i\}_{i=1}^N$

procedure PT-STEP($\{x_i\}_{i=1}^N$)
  $\{x_i^{(\text{mcmc})}\}_{i=1}^N \leftarrow$ MCMC-STEP($\{x_i\}_{i=1}^N$) ▷ e.g. Algorithm 2.1
  for i `in` 2 `to` N **do**
    $a \sim \mathcal{U}(0,1)$
    if $a < A_{\text{exch}}(x_i^{(\text{mcmc})} \leftrightarrow x_{i-1}^{(\text{mcmc})})$ **then** ▷ Exchange step
      `swap`$(x_i^{(\text{mcmc})}, x_{i-1}^{(\text{mcmc})})$

$$A_{\text{exch}}(x_T \leftrightarrow x_{T'}) = \min\,(1, \exp\left\{\frac{\tilde{p}_T(x_{T'})\tilde{p}_{T'}(x_T)}{\tilde{p}_T(x_T)\tilde{p}_{T'}(x_{T'})}\right\}) \tag{2.7}$$

Running parallel tempering involves many parallel chains at different temperatures (typically increasing with a factor of $\sqrt{2}$ starting from $T = 1$). Choosing these pairs of systems is an $\mathcal{O}(N^2)$ routine where N is the total number of parallel chains. To overcome this computational overhead, we simply pick adjacent chains for the replica exchange making this an $\mathcal{O}(N)$ routine. In practice, this approach has proven to allow enough cross-chain exchanges. This is summarized in the Algorithm 2.2 (with some notational overload).

2.3 Experiments

We first take a look at the simplest case with Hamiltonian Monte Carlo and then showcase the pathologies. In Figure 2.1 we have plotted the two-dimensional Gaussian function

$$\mathcal{N}(\mathbf{0}, \mathbf{I}) = \frac{1}{2\pi}\exp\left\{-\frac{1}{2}\mathbf{x}^T\mathbf{x}\right\} \tag{2.8}$$

We simulate the HMC Algorithm 2.1 for 2000 timesteps with $\tau = 10, \epsilon = 0.1$ and plot all

the samples obtained in Figure 2.1. HMC has no troubles sampling from this distribution as we can qualitatively visualize the typical set covered pretty well.

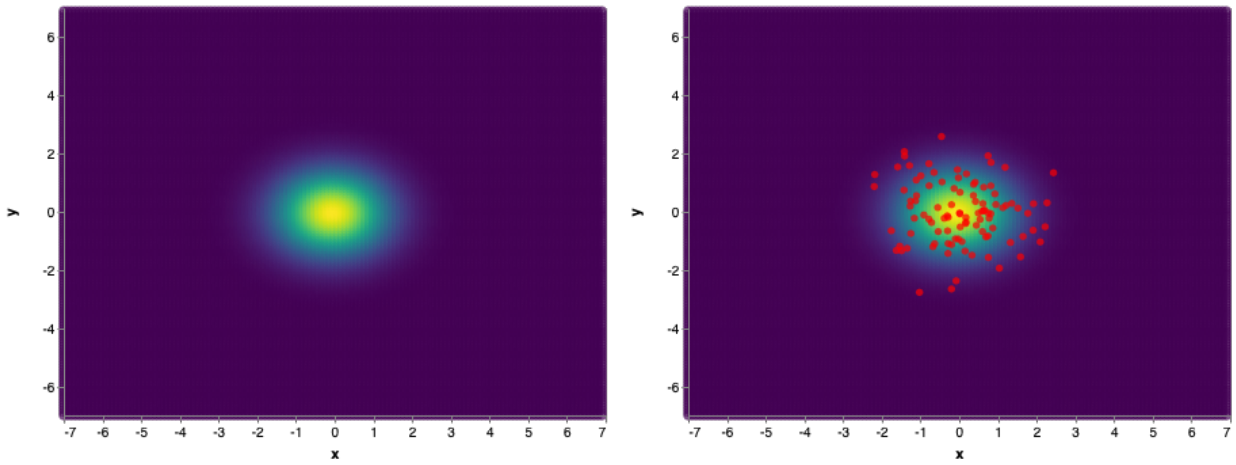


Figure 2.1: Samples from a Markov chain on $\mathcal{N}(\mathbf{0}, \mathbf{I}); \tau = 10, \epsilon = 0.1$ sub-sampled from 2000 time steps

A typical qualitative analysis involves building trace plots and seeing if the chain hovers around a small neighborhood. This hints towards convergence. In Figure 2.2, we visualize the trace plots and see that both dimensions x and y converge to the neighborhood around 0 over time which is in fact the true mean as well.

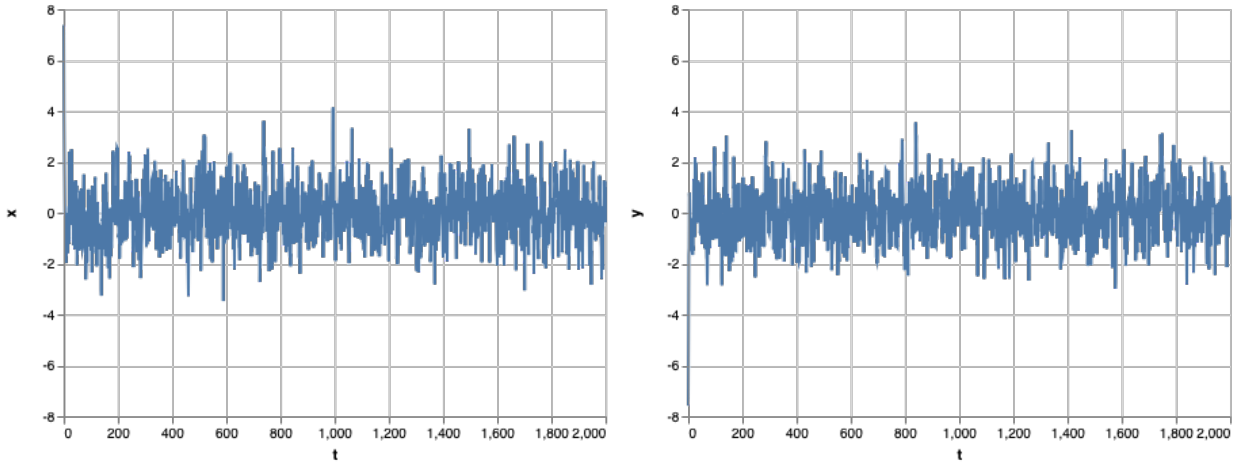


Figure 2.2: Trace plots of a converged Markov chain for $\mathcal{N}(\mathbf{0}, \mathbf{I}); \tau = 10, \epsilon = 0.1$

Building Markov chains requires careful tuning of the leapfrog integration steps τ and continuous time discretization ϵ or the chain will take too long to converge. One of the heuristics is to make sure that the product of ϵL equals 1 (Neal et al. 2011). As an example, using $\epsilon = 0.01$ makes the chain diffuse too slow through the space and under the same computational budget of 2000 timesteps, the chain does not converge just yet. This is visualized in Figure 2.3. It should be noted that in principle these chains can be run long enough and be made to converge. However, in practice we seek high quality representative samples under a constrained computational budget.

The samples generated by a Markov chain are correlated. Hence, effectively for a total of T timesteps, we do not particularly have T independent and identically distributed samples (i.i.d). Quantitatively, we measure this disparity using the *Effective Sample Size* (Gelman et al. 2013). This value is computed per dimension and the values for the previous two experiments are reported in Table 2.1.

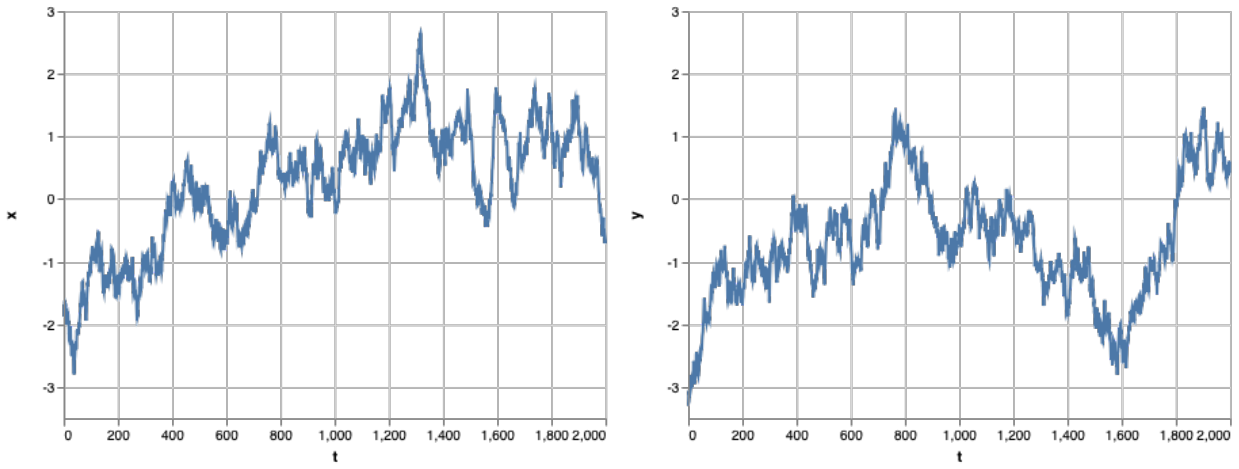


Figure 2.3: Trace plots of an unconverged Markov chain for $\mathcal{N}(\mathbf{0}, \mathbf{I}); \tau = 10, \epsilon = 0.01$

As mentioned earlier, in practice, the Markov chain will have non-zero correlation between consecutive timesteps. To minimize this side-effect, we generally throw away a few timesteps from the start as the chain may still be moving across low density regions. Additionally, to minimize correlation, we apply subsampling (or thinning) to select a sample every t^{th} time step. Table 2.1 shows the impact of such a heuristic as the *effective sample size* is a magnitude larger than the previously converged chain.

Chain State	x	y	∇ evaluations
Unconverged	3.8315	9.9608	2000
Converged	658.2034	648.7103	2000
Converged (burn-in & thinning)	1331.4050	1357.5525	5000

Table 2.1: Effective Sample Size and gradient evaluations of HMC on $\mathcal{N}(\mathbf{0}, \mathbf{I})$ with 2000 samples

While giving us high quality samples, a downside of this approach is that it is clearly wasteful and we spend larger fraction of sample for gradient evaluations than the effective representative samples.

We now consider a harder distribution in the form of a mixture of two Gaussians. The unnormalized density function is given by

$$\tilde{p}_{\text{mog2}} = \mathcal{N}\left(\begin{bmatrix}-5.0\\0.0\end{bmatrix}, 0.5 \times \mathbf{I}\right) + \mathcal{N}\left(\begin{bmatrix}5.0\\0.0\end{bmatrix}, 0.5 \times \mathbf{I}\right) \tag{2.9}$$

A few samples from Markov chain generated via HMC are show in Figure 2.4.

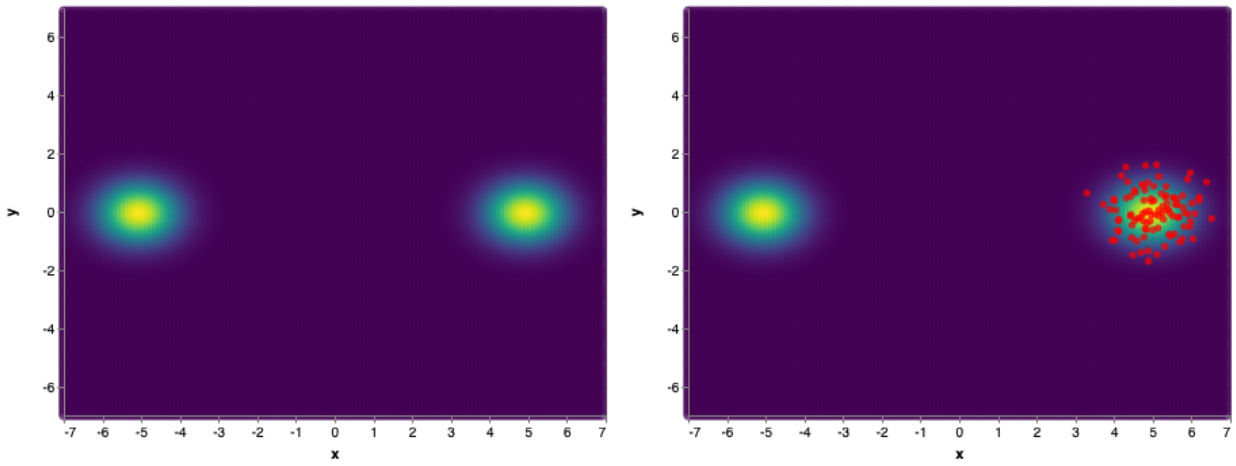


Figure 2.4: Samples from a Markov chain for $\tilde{p}_{\text{mog2}}$

We observe that the chain collapses into one of the modes and is never able to jump out from the mode. The trace plots for both dimensions of this chain are shown in Figure 2.5. To be sure, we have run this chain for 5000 time steps and the chains have converged to $x = 5$ and $y = 0$ as seen in the figure.

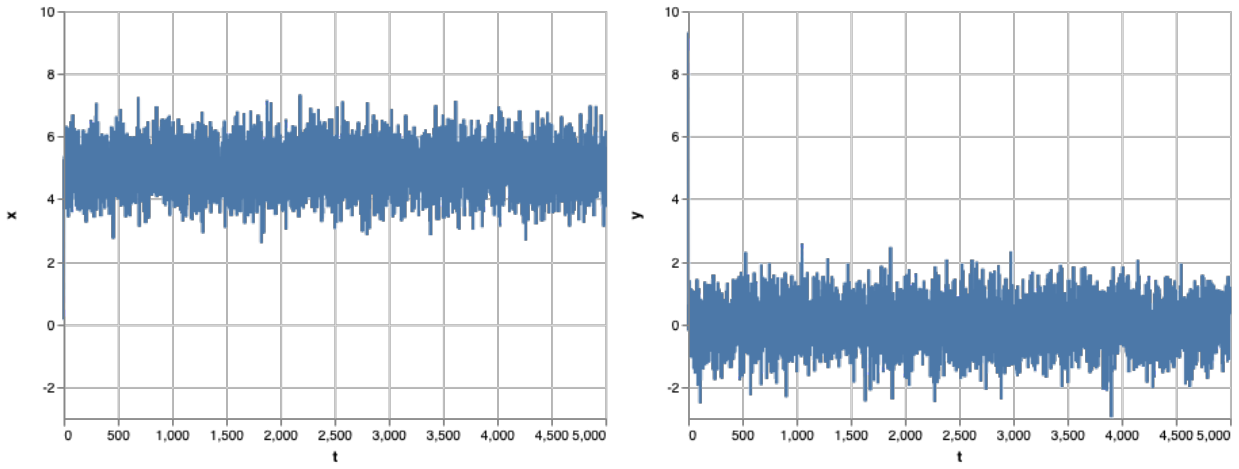


Figure 2.5: Trace plots of a Markov chain for $\tilde{p}_{\text{mog2}}$

This is a result of the usage of the gradient information while the Markov chain simulation and correct. The only way to fix this is to sample momentum large enough so as to jump over the large region of low density between the two modes. This example illustrates that in the absence of knowledge about multi-modality of the underlying density function, we might as well start believing that there are no other modes to explore as the trace plots hint towards convergence to a mode.

In the case of a mixture of six Gaussians defined below, we see that the particles mix between two close by modes and the rest of the modes remain untouched in the sampling process.

$$\mu_i = \begin{bmatrix} \cos \frac{2\pi}{6} i \\ \sin \frac{2\pi}{6} i \end{bmatrix} \tag{2.10}$$

$$\tilde{p}_{\text{mog6}} = \sum_{i=1}^{6} \mathcal{N}\left(\mu_i, 0.5 \times \mathbf{I}\right) \tag{2.11}$$

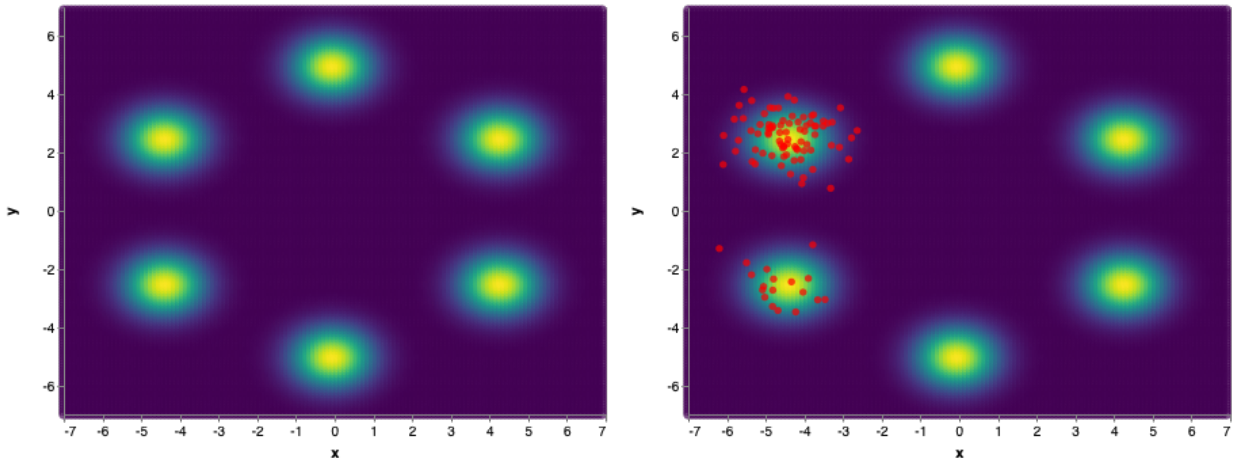


Figure 2.6: Samples from a Markov chain for $\tilde{p}_{\text{mog6}}$

A similar situation is observed in a synthetic distributions termed *Ring*5 which is a mixture of five rings. This is visualized in Figure 2.7 where we see that the samples have locked onto one of the rings. Note that in such complex geometries, trace plots become futile.

$$d_i = (||\mathbf{x}|| - i)/0.04 \tag{2.12}$$

$$\tilde{p}_{\text{ring5}} = \exp\left\{ - \min_{d \in \{d_i\}_{i=1}^{5}} d \right\} \tag{2.13}$$

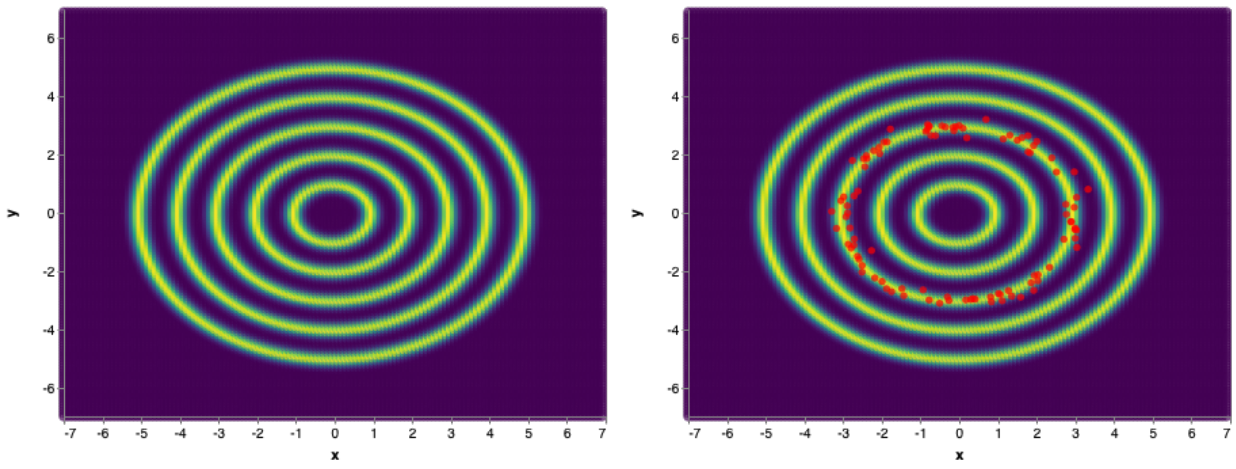
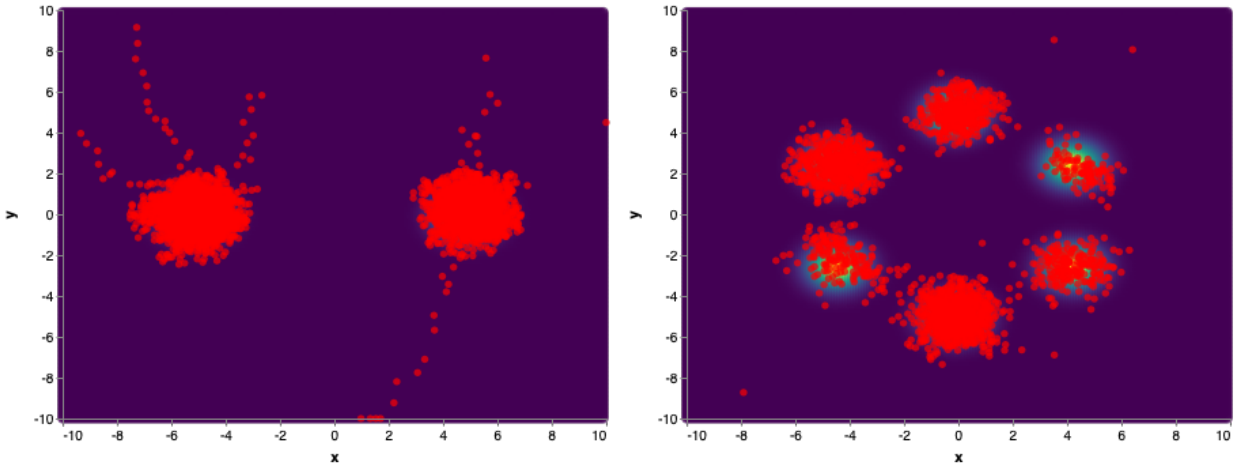


Figure 2.7: Samples from a Markov chain for $\tilde{p}_{\text{ring5}}$

One of the approaches in practice to overcome this mode collapse is to run a number of parallel chains with overdispersed initializations so that during the course of simulation, they cover enough parts of the space and each mode is visited by at least one chain. A few results from these parallel chains are shown in the upcoming figures.



(a) (b)

Figure 2.8: Samples from 10 parallel Markov chain for (a) $\tilde{p}_{\text{mog2}}$, (b) $\tilde{p}_{\text{mog6}}$

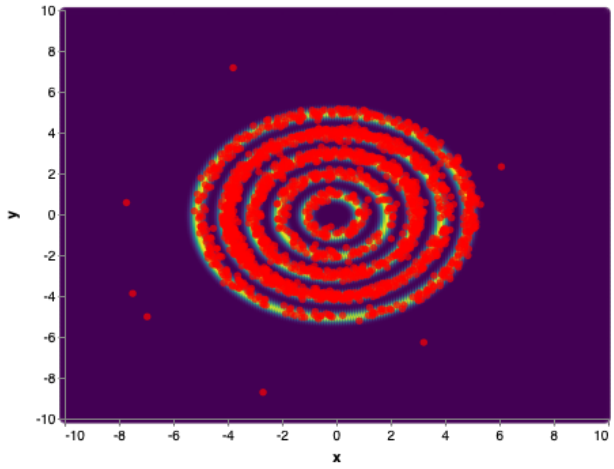


Figure 2.9: Samples from 10 parallel Markov chain for $\tilde{p}_{\text{ring5}}$

Another synthetic density function is the decaying periodic function given by a mixture of Gaussian mask applied to the periodic kernel function.

$$\tilde{p}_{\text{periodic}} = \sigma^2 \exp\left\{-\frac{2\sin^2 2\pi|x_1 - x_2|/p}{\ell^2}\right\} \tag{2.14}$$

where p is the period and ℓ determines the lengthscale function (MacKay and Mac Kay 2003).

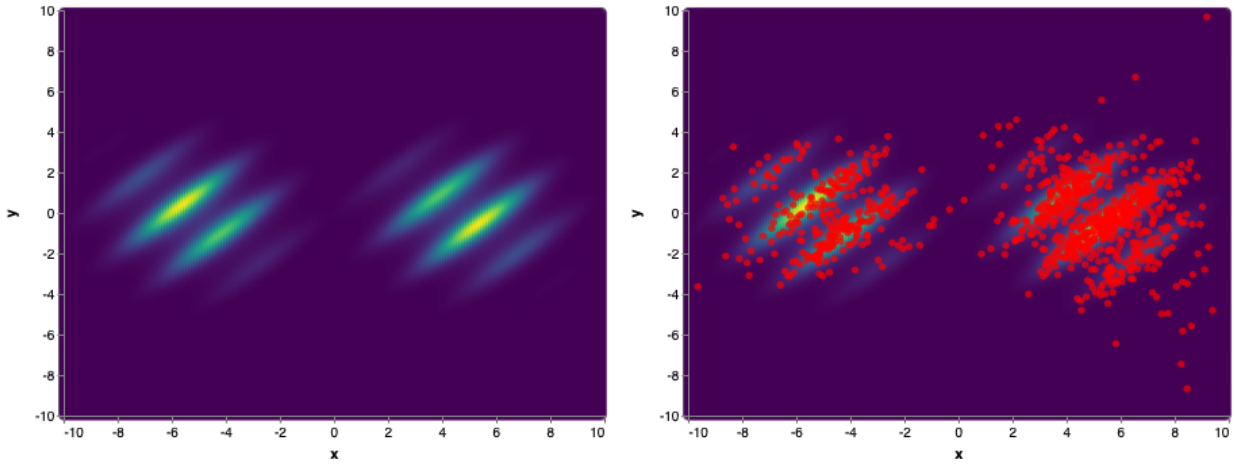


Figure 2.10: Samples from 10 parallel Markov chain for $\tilde{p}_{\text{periodic}}$; $p = 3.0$

Figure 2.10 shows how many disjoint modes are covered by the HMC sampling routine via parallel chains. A common theme to observe here is that many parallel chains need to be evaluated in these cases to get a good coverage of the typical set of the underlying distribution. There is a lot of redundancy. For instance, in Figure 2.8, we can see that 3 of the chains collapse into one mode and the remaining 2 collapse into the second mode. This is wasteful as ideally under the limitations of Hamiltionian Monte Carlo, 2 parallel chains should have been enough to cover the typical set of $\tilde{p}_{\text{mog2}}$. These challenges are alleviated in the upcoming discussion on Stein Variational Gradient Descent.

Chapter 3

Stein Variational Gradient Descent

Algorithms from the Markov Chain Monte Carlo family typically involve initializing multiple chains and running them long enough. To minimize the effects of any correlation between timesteps in the chain, we typically rely on heuristic approaches like subsampling to the chains and picking every t^{th} timestep.

An alternative family of particle-based algorithms involve simulating particles so that at the end of simulation we get representative samples from the target distribution. A recent proposal using the *Stein* criterion is known as Stein Variational Gradient Descent and we discuss it further.

3.1 Background

There are a few key ideas that make up the complete algorithm to sample from a (unnormalized) target density. We discuss them below.

3.1.1 Reproducible Kernel Hilbert Space

A function $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is called a kernel over $\mathcal{X}$. For any two points x, x' we want to define the inner product of featurized vectors $\Phi(x), \Phi(x')$ as

$$K(x, x') = \langle \Phi(x), \Phi(x') \rangle \tag{3.1}$$

for some mapping $\Phi : \mathcal{X} \to \mathbb{H}$ to a feature space called the *Hilbert Space*. By basing this formulation on top of dot products, we have essentially defined a similarity metric and hence can appeal to all the geometric niceties of angles, lengths and distances (Scholkopf and Smola 2001).

As it turns out every *Positive definite symmetric kernel* K defines a unique *Reproducing Kernel Hilbert Space* $\mathbb{H}$ with a mapping Φ. This formulation also leads to a very interesting property known as the *reproducing* property which is stated as

$$h(x) = \langle h, K(x, \cdot) \rangle \tag{3.2}$$

for all $h \in \mathbb{H}$ and $x \in \mathcal{X}$. Intuitively, this property can be interpreted as the evaluation of the function h being a linear combination of kernel evaluations over the full space $\mathcal{X}$. This is a very expressive *representer of evaluations* which takes into account the full feature space.

3.1.2 Stein's Identity

We take a detour before understanding Stein's identity. Machine Learning typically involves building a model q for an unknown true data generating distribution p. A family of measures that quantify this discrepancy between the model and the true distributions is *Integral Probability Metric* (Müller 1997). This metric simply builds an expressive family of test functions and computes the maximum discrepancy under the two distributions as

$$d(p,q) = \sup_{h \in \mathcal{H}} \left| \mathbb{E}_{X \sim q} \left[h(X) \right] - \mathbb{E}_{Y \sim p} \left[h(Y) \right] \right| \tag{3.3}$$

There are a few key elements to decode in this arrangement - the family of test functions $\mathcal{H}$ and the computation of $\mathbb{E}_{Y \sim p} \left[h(Y) \right]$. Choosing a very expressive family can lead to intractability of the supremum and only needs to be rich enough to distinguish the two distributions. Computation of the second term $\mathbb{E}_{Y \sim p} \left[h(Y) \right]$ relies on the knowledge of p which is unknown. The only way to estimate this quantity is via a Monte Carlo averaging of function evaluations over the data set. Modern variants of Generative Adversarial Networks (Goodfellow et al. 2014) like the Wasserstein GAN (Arjovsky, Chintala, and Bottou 2017) and MMD-GAN (Li et al. 2017) rely on this for computing the gradients.

To the contrary, consider the following *Stein's* operator $\mathcal{A}_p$ of a distribution p on any vector-valued function $\mathbf{f}$

$$\mathcal{A}_p \mathbf{f}(x) = \nabla_x \log p(x) \mathbf{f}(x)^T + \nabla_x \mathbf{f}(x) \tag{3.4}$$

Any smooth function $\mathbf{f}$ is considered to belong to the *Stein* class of p if

$$\int_{x \in \mathcal{X}} \nabla_x (\mathbf{f}(x) p(x)) dx = 0 \tag{3.5}$$

and this allows us to arrive at the *Stein's Identity*.

$$\mathbb{E}_p \left[\mathcal{A}_p f(x) \right] = 0 \tag{3.6}$$

Connecting this back to Equation 3.3, if we were to choose $\mathcal{H}$ such that every h belongs to the Stein class of a target distribution p, it nullifies the second term and leaves us in full control of the model q. All we need now are tractable formulations for the supremum.

3.1.3 Kernelized Stein Discrepancy

We've nullified the term containing unknown p. As a result, what we are left with $\mathbb{E}_q[h(x)]$. We can show that (Ley, Swan, et al. 2013)

$$\mathbb{E}_q[\mathcal{A}_p h(x)] = \mathbb{E}_q\left[(\nabla_x \log p(x) - \nabla_x \log q(x))h(x)^T\right] \tag{3.7}$$

The magnitude of this quantity relates to how different p and q are. Intuitively, this can be viewed as the function evaluation being a difference between the score functions of the two distributions being compared. Further Gorham and Mackey 2015 showed

$$\mathbb{E}_q\left[(\nabla_x \log p(x) - \nabla_x \log q(x))h(x)^T\right] = \mathbb{E}_q[\text{trace } (\mathcal{A}_p h(x))] \tag{3.8}$$

Combining these observations, we arrive at the *Stein Discrepancy* between two continuous distributions q and p over a family of test functions $\mathcal{F}$ as

$$\mathbb{S}(q,p) = \max_{\Phi \in \mathcal{F}} \left\{[\mathbb{E}_q \text{trace } (\mathcal{A}_p \Phi(x))]^2\right\} \tag{3.9}$$

Choosing the family of functions $\mathcal{F}$ to be in the unit norm RKHS allows for the optimization to have a closed-form solution (Liu, Lee, and Jordan 2016; Liu and D. Wang 2016).

$$\mathbb{S}(q,p) = \max_{\Phi \in \mathcal{H}^d} \left\{[\mathbb{E}_q \text{trace } (\mathcal{A}_p \Phi(x))]^2 \text{ s.t. } ||\Phi||_{\mathcal{H}^d} \leq 1\right\} \tag{3.10}$$

for the Hilbert space $\mathcal{H}^d$ induced by the kernel k. The optimal solution is given by

$$\Phi(x) = \Phi^\star(x)//||\Phi^\star(x)||_{\mathcal{H}^d} \tag{3.11}$$

$$\text{where, } \Phi^\star(\cdot) = \mathbb{E}_q[\mathcal{A}_p k(x,\cdot)] \tag{3.12}$$

This makes the *Kernelized Stein Discrepancy* to be $\mathbb{S}(q,p) = ||\Phi^\star||^2_{\mathcal{H}^d}$. The RBF kernel is in the Stein class of all smooth densities on $\mathcal{X} = \mathbb{R}^d$.

3.2 SVGD

We finally arrive at the idea of *Stein Variational Gradient Descent* (SVGD) which shows that the *Kernelized Stein Discrepancy* can be related to the gradient of the KL divergence between our modeled distribution q and the true underlying distribution p (Liu and D. Wang 2016).

Given an identity perturbation $\mathbf{T}(x) = x + \epsilon\phi(x)$, for $x \sim q(x)$ we have a one-step normalizing flow leading to $z = \mathbf{T}(x) \sim q_{[\mathbf{T}]}(x)$ and because of Liu and D. Wang 2016

$$\nabla_\epsilon KL(q_{[\mathbf{T}]}||p)\Big|_{\epsilon=0} = -\mathbb{E}_q\left[\text{trace }(\mathcal{A}_p\phi(x)\right] \tag{3.13}$$

For the case of *Kernelized Stein Discrepancy*, we can henceforth see that the optimal perturbation direction is the steepest descent on the KL-divergence in unit-norm balls of $\mathcal{H}^d$. Hence, at each step of the transform, we are decreasing the KL-divergence by a factor of $\epsilon\mathbb{S}(q,p)$. The complete direction is therefore given by Equation 3.12 as

$$\phi^\star = \mathbb{E}_{x\sim q}\left[k(x,\cdot)\nabla_x \log p(x) + \nabla_x k(x,\cdot)\right] \tag{3.14}$$

In practice, we can make a Monte Carlo estimate of $\phi^\star$ starting from a set of initial particles. Additionally, it can also be seen that we can replace p with an unnormalized density function $\tilde{p}$ as the score function does not depend on the normalizing constant. A step in SVGD is summarized in Algorithm 3.1.

From the perspective of communication, we see that the kernel induces interaction between particles where the second term encourages diversity. If we only have one particle and a

Algorithm 3.1 SVGD Step

Input: An unnormalized probability density function $\tilde{p}$, a set of N particles at time t $\left\{x_i^{(t)}\right\}_{i=1}^{N}$

1: **procedure** SVGD-STEP($\left\{x_i^{(t)}\right\}_{i=1}^{N}$)

2: **for** i `in` 1 `to` N **do**

3: $\phi_i^{(t)} = \frac{1}{N}\sum_{j=1}^{N}\left[k(x_j^{(t)}, x_i)\nabla_{x_j^{(t)}} \log p(x_j^{(t)}) + \nabla_{x_j^{(t)}} k(x_j^{(t)}, x_i)\right]$

4: $x_i^{(t+1)} = x_i^{(t)} + \epsilon\phi_i^{(t)}$

kernel where $\nabla k(x, x) = 0$, we can see that Equation 3.12 reduces to the classic MAP estimate (Murphy 2012). Another important advantage of this approach is that we don't need explicit form for the modeled distribution to compute normalizing flow, which otherwise would make the log-determinant of the inverse Jacobian which can be computationally prohibitive for expressive transforms like neural networks. We see these behaviors in the experiments.

3.3 Experiments

We first do a similar visual analysis as for distributions from experiments in §2.3 but this time using *Stochastic Variational Gradient Descent*. All the experiments use the *radial basis function* kernel.

$$k(x, x') = \exp\left\{\frac{||x - x'||^2}{\sigma^2}\right\} \tag{3.15}$$

The bandwidth was adaptively adjusted during the simulation of the differential equation from Algorithm 3.1 via the median heuristic (Liu and D. Wang 2016) as

$$\sigma^2 = \frac{\alpha^2}{2\log(N+1)} \tag{3.16}$$

where α is the median distance among all pairwise distances of the N particles in motion. This heuristic allows us to avoid a situation where the gradients with respect to one of the particles tend to zero and effectively limits the gradient information from that particle. Additionally, to stabilize the gradient flow, we use adaptive gradients via Adam (Kingma and Ba 2014) while stepping through the differential equation. Quite naturally, we can replace it with any other adaptive gradient method.

We first visualize the standard normal $\mathcal{N}(\mathbf{0}, \mathbf{I})$ distribution and simulate SVGD for 150 time steps with time discretization of $\epsilon = 0.1$.

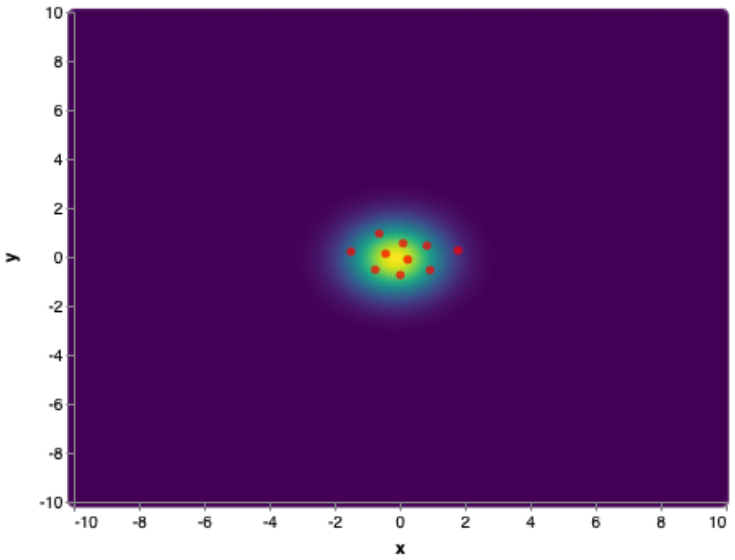
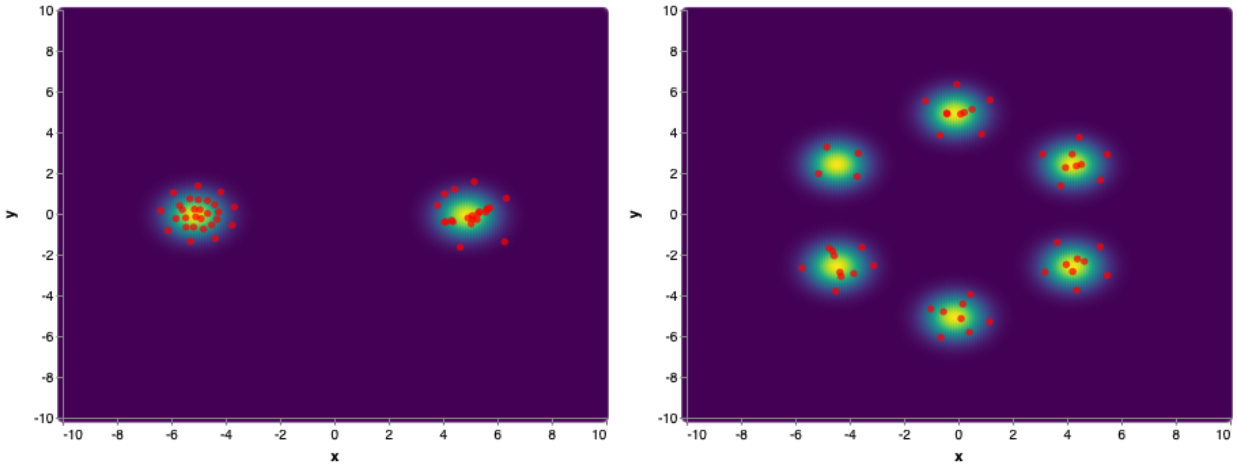


Figure 3.1: Samples using 10 Stein particles for $\mathcal{N}(\mathbf{0}, \mathbf{I})$

One of the observations to note here is that the particles are well spread out from each other over the typical set of the underlying density function. This is attributed to the repulsive kernel term in Equation 3.14. This observation remains consistent in few density functions visualized next though with more particles for enough coverage of the typical set.



(a) (b)

Figure 3.2: Samples using 50 Stein particles for (a) $\tilde{p}_{\text{mog2}}$, (b) $\tilde{p}_{\text{mog6}}$

As we see in Figure 3.2, SVGD doesn't typically suffer from the mode lock-in contrary to what see see in Figure 2.4 and 2.6. We further see that SVGD can generalize without much fine-tuning to every more complex distributions with plenty of modes.

For instance, consider a mixture of 25 Gaussians whose density function is given by

$$\tilde{p}_{\text{mog25}} = \sum_{i=-2}^{2} \sum_{j=-2}^{2} \mathcal{N}(\begin{bmatrix} i \\ j \end{bmatrix}, 0.1 \times \mathbf{I}) \tag{3.17}$$

We simulate a total of 300 particles and use the Stein gradient update which includes the kernel repulsive term and see in Figure 3.3 that SVGD is able to cover the typical sets of all the modes to a reasonable level.

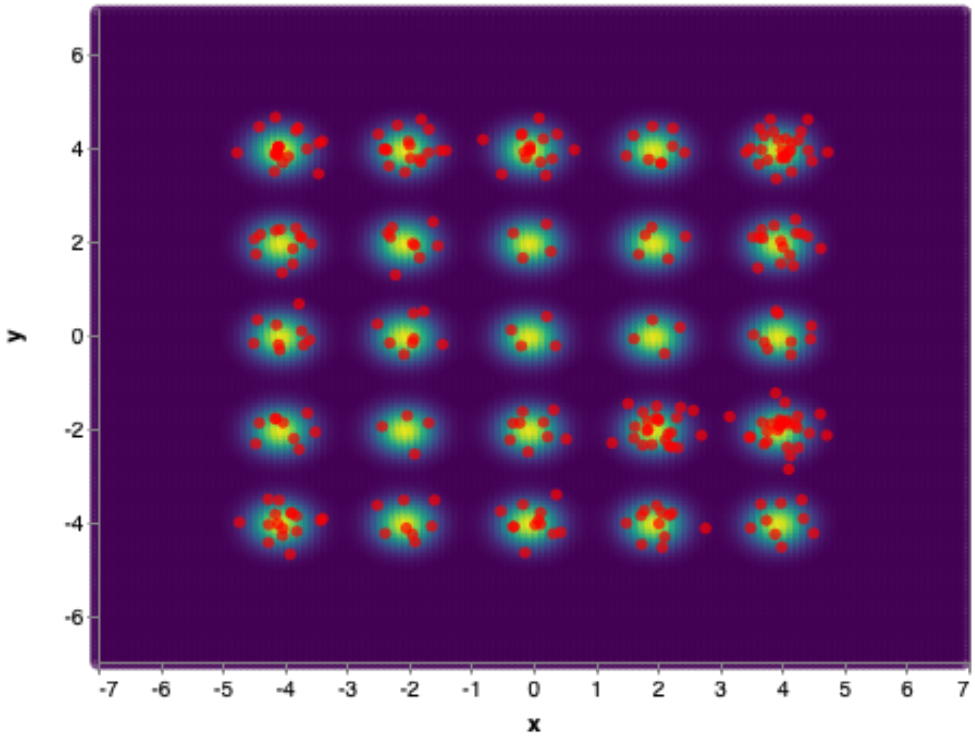
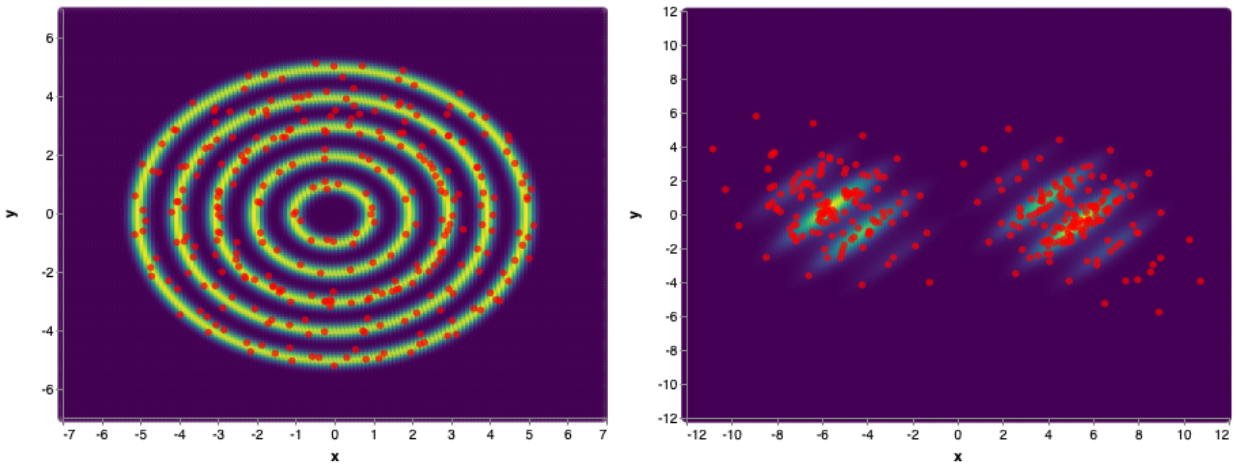


Figure 3.3: Samples using 300 Stein particles for $\tilde{p}_{\text{mog25}}$

SVGD is also able to generalize this across more complex geometries like those of $\tilde{p}_{\text{ring5}}$ and $\tilde{p}_{\text{periodic}}$ visualized in Figure 3.4.



(a) (b)

Figure 3.4: Samples using 300 Stein particles for (a) $\tilde{p}_{\text{ring5}}$, (b) $\tilde{p}_{\text{periodic}}$

A consistent theme in these visualizations is to observe the fact that the particles communicate with each other via gradient of the pairwise kernel evaluations. This repulsive term prevents particles from collapsing into one mode. Unlike, HMC where we used parallel chains and the virtue of stochasticity to ensure we cover all modes, SVGD uses a deterministic gradient evaluation on the set of simulated particles to disperse in the best possible manner. Effectively, as we've seen from theory in Equation 3.13, it tries to minimize the KL-divergence between the transformed distribution and the target density in the RKHS space induced by the kernel.

3.3.1 Maximum Mean Discrepancy

To quantitatively evaluate and compare SVGD with HMC, we introduce *Maximum Mean Discrepancy* (MMD).

Suppose we are given two sets of samples from distributions P as $X = \{x_i\}_{i=1}^{N}$ and Q as $Y = \{y_j\}_{j=1}^{M}$, one of the simplest ways of comparing whether the two distributions are the same is to compare all possible statistics on the two sets of samples (Gretton et al. 2012). This is in fact another Intergral Probability Metric (Müller 1997) that operates in the RKHS space as well. Empirically, MMD with a statistic of interest *phi* is given by

$$\mathcal{L}_{\text{MMD}} = \left|\left| \frac{1}{N}\sum_{i=1}^{N}\phi(x_i) - \frac{1}{M}\sum_{j=1}^{M}\phi(y_j) \right|\right| \tag{3.18}$$

This can be rewritten in the form of dot products as

$$\mathcal{L}_{\text{MMD}} = \frac{1}{N^2}\sum_{i=1}^{N}\sum_{i'=1}^{N}\phi(x_i)^T\phi(x_{i'}) + \frac{1}{M^2}\sum_{j=1}^{M}\sum_{j'=1}^{M}\phi(x_j)^T\phi(y_{j'}) - \frac{2}{MN}\sum_{i=1}^{N}\sum_{j=1}^{M}\phi(x_i)^T\phi(y_j) \tag{3.19}$$

This dot product form naturally alludes to the usage of kernel trick where we are effectively

lifting the samples into the *Reproducible Kernel Hilbert Space* induced by a kernel K as

$$\mathcal{L}_{\text{MMD}} = \frac{1}{N^2}\sum_{i=1}^{N}\sum_{i'=1}^{N} K(x_i, x_{i'}) + \frac{1}{M^2}\sum_{j=1}^{M}\sum_{j'=1}^{M} K(y_j, y_{j'}) - \frac{2}{MN}\sum_{i=1}^{N}\sum_{j=1}^{M} K(x_i, y_j) \tag{3.20}$$

Gretton et al. 2012 show that asymptotically $\mathcal{L}_{\text{MMD}}$ is only equal to zero if and only if $P = Q$. Another interesting result is that minimizing MMD under the feature expansion of an RBF kernel via a Taylor series expansion is equivalent to minimizing the distance between all the moments of two distributions. We compare MMD between the samples generated via HMC and SVGD to the true samples.

We first compare performance on $\tilde{p}_{\text{mog2}}$ in Figure 3.5. In this chart, we compare a single chain HMC, 10 parallel chains of HMC and 100 particles in SVGD. We compute the *Kernelized Maximum Mean Discrepancy* with an RBF kernel using bandwidth of 0.5 for finer comparison. Larger values of bandwidth may not yield perceptible differences as the kernel becomes smoother. We can see that SVGD performs better than both the HMC and HMC with parallel chains. The single chain HMC quite expectedly plateaus as it locks into one mode.

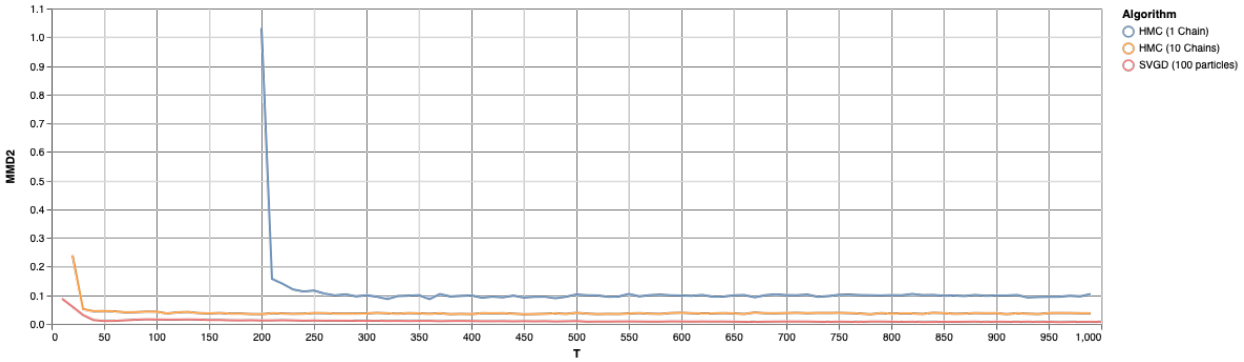


Figure 3.5: $\mathcal{L}_{\text{MMD}}$ improvement over time steps for $\tilde{p}_{\text{mog2}}$

Figure 3.6 shows a similar analysis for $\tilde{p}_{\text{mog6}}$ where the differences are much more pronounced as the geometry is more complicated to cover.

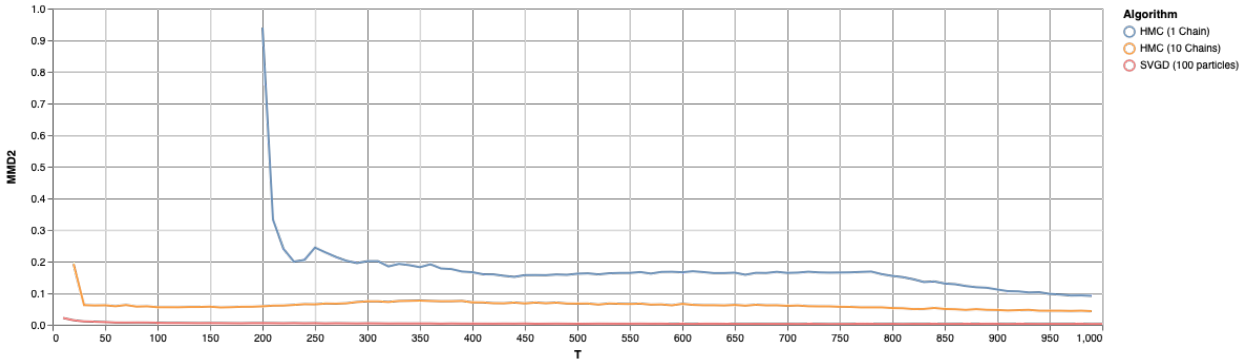


Figure 3.6: $\mathcal{L}_{\text{MMD}}$ improvement over time steps for $\tilde{p}_{\text{mog6}}$

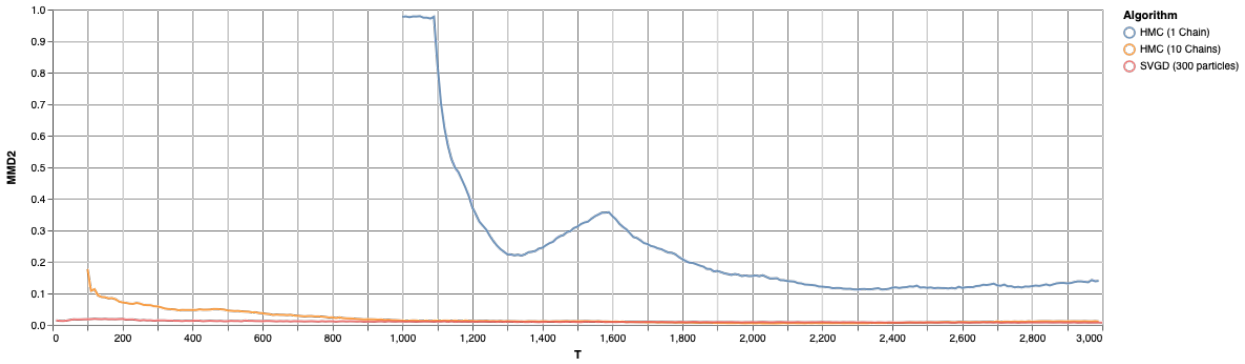


Figure 3.7: $\mathcal{L}_{\text{MMD}}$ improvement over time steps for $\tilde{p}_{\text{mog25}}$

In these figures, it should be noted that we use the thinning heuristic to minimize correlations between consecutive particles in a chain just like one would when using these samples in practice. We effectively end up doing more computation than we use samples from in practice which is unlike SVGD which by design manipulates particles to make them representative of the underlying density function and seems to be much more efficient.

One of the key pain points of SVGD is that the kernel evaluations are quadratically expensive. While we get away in these experiments with relatively small number of particles, it remains to be seen whether certain geometries and higher dimensions need more particles to work effectively or not. Another downside of both HMC and SVGD is the simulation of the

differential equations. In an application where the latency of generation is crucial, this can be unacceptable. In the next chapter, we look at how we can use neural networks to amortize the generative process by learning the dynamics into neural networks for fast sampling.

Chapter 4

Amortized Dynamics with Communication

We have seen how a set of parallel chains are used by HMC to sample from different parts of the space, albeit redundant. HMC typically requires us to apply heuristics like burn-in and thinning to make sure the chains have mixed well enough before we use them as representative samples. On the other hand we see via SVGD how we can do away with such heuristics and rely on communication via kernel evaluations for sampling. However, one of the concerns with SVGD is the costly $\mathcal{O}(N^2)$ kernel computations and simulation of the differential equations to convergence. There is scope to build faster samplers which can bypass this element of repeatedly simulating differential equations for real-time usage.

4.1 Amortizing the Dynamics

A simple choice to amortize the dynamics of a differential equation is to simply consider the input-output relationship between the particles and then approximate them via an expressive parametric function f as $x = f_\theta(z)$ parametrized by θ. f could be a neural network. However,

this formulation of the problem is particularly hard and slow to train [1].

D. Wang and Liu 2016 propose to adjust the parameters θ in an incremental manner by following the Stein Variational Gradient Descent. Specifically, consider sampling a noise particle z and the transform $x = f_\theta(z)$. The Stein Variational Gradient provides the direction of steepest descent in the RKHS which can minimize the KL divergence to some target density $\tilde{p}$ as in Equation 3.13. This can be represented by the transform $x' = x + \epsilon\Delta x$ where Δx represents the update previously seen in Equation 3.12 and Equation 3.14. This new particle is the target we would like our estimator to achieve and we can choose a metric to minimize over the full sample set, for instance the Euclidean distance. As a result we get the following formulation of the learning problem over a collection of M particles as

$$\theta^\star = \underset{\theta\in\Theta}{\operatorname{argmin}} \sum_{i=1}^{M} ||f_\theta(z_i) - x_i'||_2^2 \tag{4.1}$$

where $x_i' = x_i + \epsilon\Delta x_i$ and Δx_i is the same empirical update as in Algorithm 3.1. At the end of this problem f_θ becomes our fast amortized sampler for some underlying density function $\tilde{p}$. Under the assumption of infinitesimal ϵ and Taylor series expansion of f around the current parameter estimate $\theta^{(t)}$, we get

$$||f_\theta(z_i) - x_i'||_2^2 \approx ||\partial_\theta f_{\theta^{(t)}}(z_i)(\theta - \theta^{(t)}) - \epsilon\Delta x_i||_2^2 \tag{4.2}$$

We can now use gradient descent and apply a one step update starting in the limit $\theta \to \theta^{(t)}$ and arrive at the final iterative approximation of the parameter update step as

$$\theta^{(t+1)} \leftarrow \theta^{(t)} + \epsilon \sum_{i=1}^{M} \partial_\theta f_{\theta^{(t)}}(z_i)\Delta x_i \tag{4.3}$$

We summarize this in Algorithm 4.1 with some notational overloading for brevity.

[1] experiments using *maximum mean discrepancy* as the criterion did not bear fruit

Algorithm 4.1 Amortized SVGD

Input: An unnormalized probability density function $\tilde{p}$, initial parameters $\theta^{(0)}$ for estimator f, proposal density function q (typically $\mathcal{N}(\mathbf{0}, \mathbf{I})$), kernel function k, batch size M

procedure AMORTIZED-SVGD

  $\{z_i\}_{i=1}^{M} \sim q$

  $\{x_i\}_{i=1}^{M} = f_{\theta^{(t)}}\left(\{z_i\}_{i=1}^{M}\right)$

  for i `in` 1 `to` M **do**

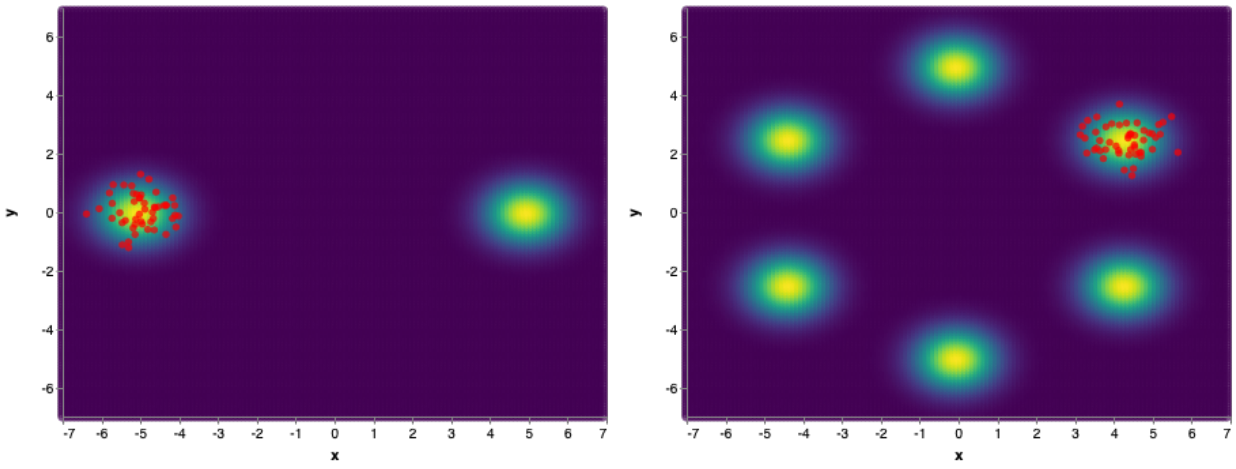
    $\Delta x_i = \frac{1}{M}\sum_{j=1}^{M}\left[k(x_j, x_i)\nabla_{x_j}\log p(x_j) + \nabla_{x_j}k(x_j, x_i)\right]$

  $\theta^{(t+1)} \leftarrow \theta^{(t)} + \epsilon\sum_{i=1}^{M}\partial_\theta f_{\theta^{(t)}}(z_i)\Delta x_i$

4.1.1 Experiments

We now see some results using Amortized SVGD from Algorithm 4.1. Our setup for the densities remains similar to experiments in Section 2.3 and Section 3.3.

We use a simple neural network with 2-hidden layers of size 100 with *tanh* non linearity[2]. Each network was trained using *Adam* with a learning rate of 0.001 (Kingma and Ba 2014).

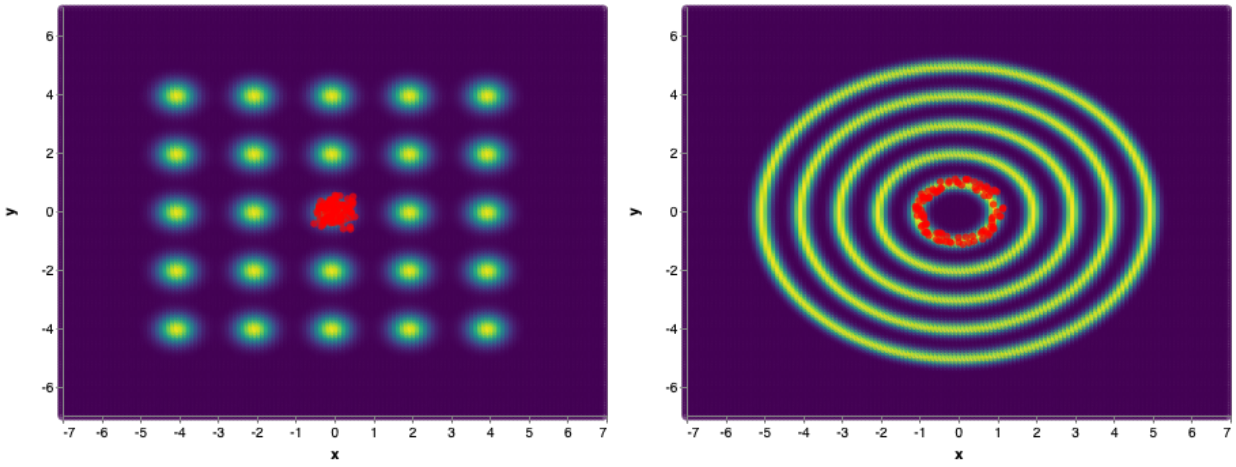


(a) (b)

Figure 4.1: Mode collapse in samples from MLP trained via Amortized SVGD (a) $\tilde{p}_{\text{mog2}}$, (b) $\tilde{p}_{\text{mog6}}$

[2] *relu* also achieved similar outputs

Samples generated from the network and are plotted in Figure 4.1. It appears that the neural network suffers from mode collapse and is not able to model the multiple modes. A similar observation is made with other density functions as well.



(a) (b)

Figure 4.2: Mode collapse in samples from MLP trained via Amortized SVGD (a) $\tilde{p}_{\text{mog25}}$, (b) $\tilde{p}_{\text{ring5}}$

The parameters learned lead to fairly stable results, albeit with mode collapse. Further training does not improve or deteriorate the quality of samples from what has been achieved.

4.2 Learning to communicate

The two key components in Algorithm 4.1 that we rely on for quality are the inductive biases encoded in the parametric architecture of function transform f and the kernel k. The kernel is particularly influential as it induces diversity by acting as the additional repulsive force in the standard likelihood maximization problem.

We hypothesize that this communication via the kernel evaluations can be replaced by expressive *graph neural networks* (Zhou et al. 2018). Graph Neural Networks are a potential candidate to model the communication as they help model a message passing interface be-

tween nodes in a graph.

We specifically formulate the problem of amortizing the dynamics of SVGD as in the previous section as problem of learning graph representations. We model the particles $\{x_i\}_{i=1}^N$ of interest as nodes in a complete graph. The graph is complete to allow communication between each pair of nodes. Each node updates its state with respect to the neighbors using the following transformation equations for each node of a graph $x_i \in G$

$$h_i^{(l)} = f_\theta(z_i^{(l)}) \tag{4.4}$$

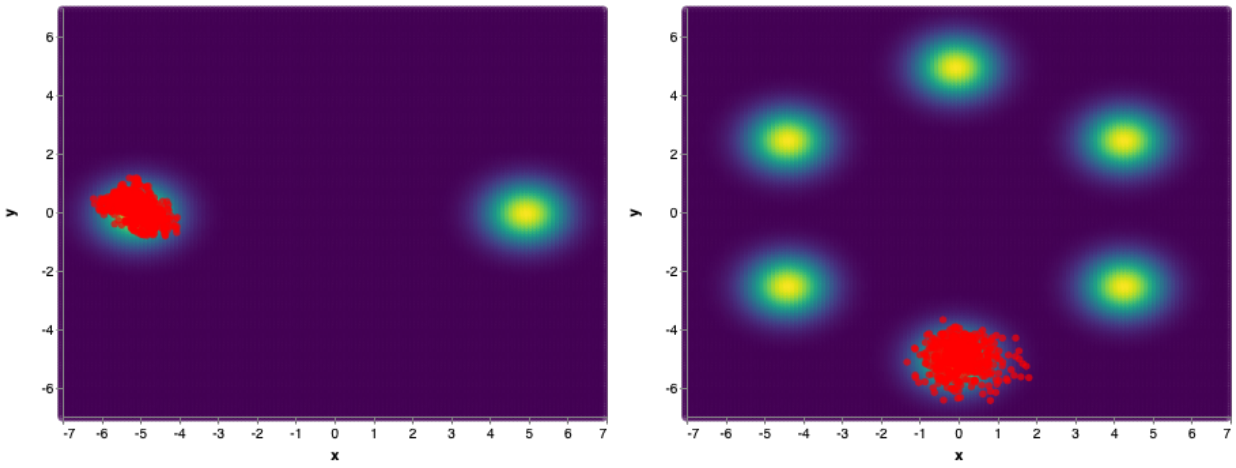
$$\alpha_i = \text{softmax}(f_\phi(z_i^{(l)}, \left\{z_j^{(l)}\right\}_{j \in \mathcal{N}(i)})) \tag{4.5}$$

$$h_i^{(l+1)} = \sum_{j \in \mathcal{N}(i)} \alpha_{ij} z_j^{(l)} \tag{4.6}$$

Effectively, we first build the embedding $z_i \in \mathbb{R}^h$ for each node $x_i \in \mathbb{R}^d$ using f_θ, using a message passing routine, we build attention $\alpha_i \in \mathbb{R}^{|\mathcal{N}(i)|}$ over the set of neighbors $\mathcal{N}(i)$ using f_ϕ and then use a linear combination of all the messages, which in this case is simply the embeddings of all the neighbors to compute the final state of the node $h_i \in \mathbb{R}^d$. This is effectively equivalent to a Graph Attention Network (Veličković et al. 2017). We use multiple-heads and multiple layers l of such attention modules as described by equations above and combine the outputs of all the heads via mean aggregation. Additionally, to break symmetries in a complete graph, we apply neighborhood sub-sampling to the communication channel between pair of nodes. This is effectively equivalent to applying dropout before applying `softmax` in Equation 4.5.

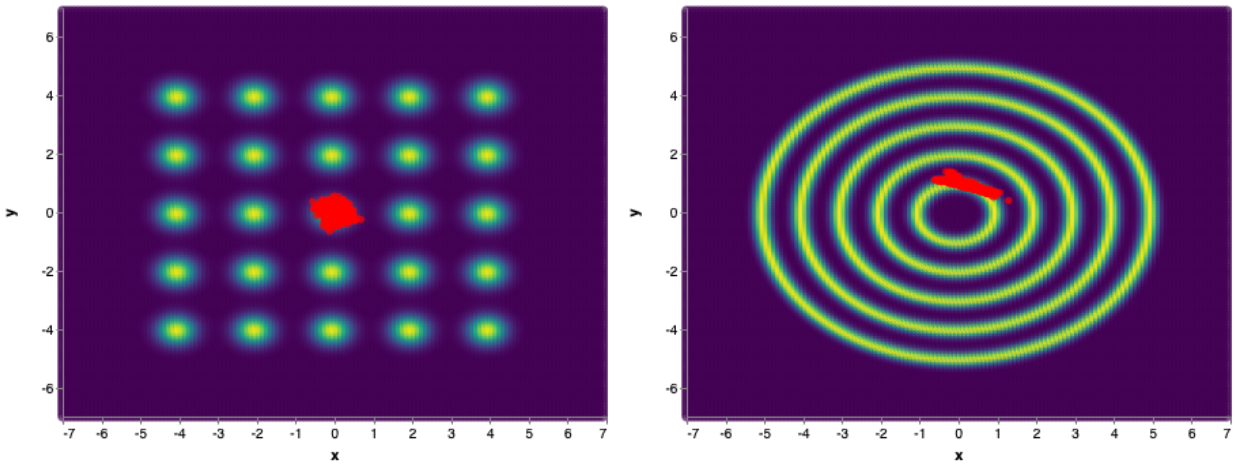
4.2.1 Experiments

So far, using this approach, we have been able to train stable networks however these still suffer from mode collapse. Further work is needed to get this to achieve better coverage of multiple modes.



(a) (b)

Figure 4.3: Mode collapse in samples from Graph Attention Sampler (a) $\tilde{p}_{\text{mog2}}$, (b) $\tilde{p}_{\text{mog6}}$



(a) (b)

Figure 4.4: Mode collapse in samples from Graph Attention Sampler (a) $\tilde{p}_{\text{mog25}}$, (b) $\tilde{p}_{\text{ring5}}$

Conclusion and Future Work

The comparison of HMC and SVGD shows us how communication can help us improve our sampling schemes using just the gradient information of a continuous density function. Using single particle *Langevin* dynamics as in HMC is not particularly efficient even though it utilizes the gradient information of the density surface. HMC finds it extremely improbable to jump across modes in a single chain over the low density surfaces. SVGD on the other hand benefits from repulsion induced by the kernel evaluations however comes at additional computation burded of $\mathcal{O}(N^2)$ kernel evaluations. Amortization of these dynamics into an expressive neural network can allow us to sample fast without repeatedly simulating long trajectories of HMC or SVGD. Current architectures remain ineffective in their vanilla form to be able to model the dynamics and hence need stronger inductive biases to be able to perform. The overarching idea of using communication between particles however clearly remains a promising direction to pursue to prepare modern samplers for real-time usage.

Bibliography

[1] Martin Arjovsky, Soumith Chintala, and Léon Bottou. "Wasserstein generative adversarial networks". In: *International Conference on Machine Learning*. 2017, pp. 214–223.

[2] Peter W. Battaglia et al. *Relational inductive biases, deep learning, and graph networks*. 2018. arXiv: `1806.01261 [cs.LG]`.

[3] Michael Betancourt. "A conceptual introduction to Hamiltonian Monte Carlo". In: *arXiv preprint arXiv:1701.02434* (2017).

[4] Simon Duane et al. "Hybrid monte carlo". In: *Physics letters B* 195.2 (1987), pp. 216–222.

[5] David J Earl and Michael W Deem. "Parallel tempering: Theory, applications, and new perspectives". In: *Physical Chemistry Chemical Physics* 7.23 (2005), pp. 3910–3916.

[6] Andrew Gelman et al. *Bayesian data analysis*. Chapman and Hall/CRC, 2013.

[7] Mark Girolami and Ben Calderhead. "Riemann manifold langevin and hamiltonian monte carlo methods". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 73.2 (2011), pp. 123–214.

[8] Ian Goodfellow et al. "Generative adversarial nets". In: *Advances in neural information processing systems*. 2014, pp. 2672–2680.

[9] Jackson Gorham and Lester Mackey. "Measuring sample quality with Stein's method". In: *Advances in Neural Information Processing Systems*. 2015, pp. 226–234.

[10] Arthur Gretton et al. “A kernel two-sample test”. In: *Journal of Machine Learning Research* 13.Mar (2012), pp. 723–773.

[11] Tuomas Haarnoja et al. *Reinforcement Learning with Deep Energy-Based Policies*. 2017. arXiv: 1702.08165 [cs.LG].

[12] Diederik P Kingma and Jimmy Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014).

[13] Christophe Ley, Yvik Swan, et al. “Stein’s density approach and information inequalities”. In: *Electronic Communications in Probability* 18 (2013).

[14] Chun-Liang Li et al. “Mmd gan: Towards deeper understanding of moment matching network”. In: *Advances in Neural Information Processing Systems*. 2017, pp. 2203–2213.

[15] Qiang Liu, Jason Lee, and Michael Jordan. “A kernelized Stein discrepancy for goodness-of-fit tests”. In: *International Conference on Machine Learning*. 2016, pp. 276–284.

[16] Qiang Liu and Dilin Wang. “Stein variational gradient descent: A general purpose bayesian inference algorithm”. In: *Advances In Neural Information Processing Systems*. 2016, pp. 2378–2386.

[17] David JC MacKay and David JC Mac Kay. *Information theory, inference and learning algorithms*. Cambridge university press, 2003.

[18] Nicholas Metropolis et al. “Equation of state calculations by fast computing machines”. In: *The journal of chemical physics* 21.6 (1953), pp. 1087–1092.

[19] Alfred Müller. “Integral probability metrics and their generating classes of functions”. In: *Advances in Applied Probability* 29.2 (1997), pp. 429–443.

[20] Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012. ISBN: 9780262018029.

[21] Radford M Neal et al. “MCMC using Hamiltonian dynamics”. In: *Handbook of markov chain monte carlo* 2.11 (2011), p. 2.

[22] OpenAI. *OpenAI Five*. https://blog.openai.com/openai-five/. 2018.

[23] Bernhard Scholkopf and Alexander J Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2001.

[24] David Silver et al. "Mastering the game of go without human knowledge". In: *Nature* 550.7676 (2017), p. 354.

[25] Jascha Sohl-Dickstein, Mayur Mudigonda, and Michael DeWeese. "Hamiltonian Monte Carlo Without Detailed Balance". In: *Proceedings of the 31st International Conference on Machine Learning*. Ed. by Eric P. Xing and Tony Jebara. Vol. 32. Proceedings of Machine Learning Research 1. Bejing, China: PMLR, 2014, pp. 719–726. URL: http://proceedings.mlr.press/v32/sohl-dickstein14.html.

[26] Sainbayar Sukhbaatar. "Elements of Intelligence: Memory, Communication and Intrinsic Motivation". PhD thesis. New York University, 2018.

[27] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[28] Robert H. Swendsen and Jian-Sheng Wang. "Replica Monte Carlo Simulation of Spin-Glasses". In: *Phys. Rev. Lett.* 57 (21 Nov. 1986), pp. 2607–2609. DOI: 10.1103/PhysRevLett.57.2607. URL: https://link.aps.org/doi/10.1103/PhysRevLett.57.2607.

[29] Petar Veličković et al. "Graph attention networks". In: *arXiv preprint arXiv:1710.10903* (2017).

[30] Oriol Vinyals et al. *AlphaStar: Mastering the Real-Time Strategy Game StarCraft II*. https://deepmind.com/blog/alphastar-mastering-real-time-strategy-game-starcraft-ii/. 2019.

[31] Dilin Wang and Qiang Liu. "Learning to draw samples: With application to amortized mle for generative adversarial learning". In: *arXiv preprint arXiv:1611.01722* (2016).

[32] Jie Zhou et al. *Graph Neural Networks: A Review of Methods and Applications*. 2018. arXiv: 1812.08434 [cs.LG].