Robust and Scalable Algorithms for Bayesian Nonparametric Machine Learning

Dissertation

Presented in Partial Fulfillment of the Requirements for the Degree Doctor of Philosophy in the Graduate School of The Ohio State University

By

Anirban Roychowdhury, M.S.

Graduate Program in Computer Science and Engineering

The Ohio State University

2017

Dissertation Committee:

Srinivasan Parthasarathy, Advisor

Brian J. Kulis

Mikhail Belkin

Huan Sun

Abstract

Bayesian nonparametric techniques provide a rich set of tools for modeling complex probabilistic machine learning problems. However the richness comes at the cost of significant complexity of learning and inference for large scale datasets, in addition to an orthogonal set of challenges related to algorithm convergence and correctness. In this dissertation we address these issues by developing a variety of novel methods using concepts from established machine learning methodologies as well as fields like statistical physics and differential geometry.

First, we develop fast inference algorithms for sequential models with Bayesian nonparametric priors using small-variance asymptotics, an emerging technique for obtaining scalable combinatorial algorithms from rich probabilistic models. We derive a "hard" inference algorithm analogous to k-means for the standard hidden Markov model by making particular variances in the model tend to zero, and extend the analysis to the Bayesian nonparametric case, yielding a simple, scalable, and flexible algorithm for discrete-state sequence data with a non-fixed number of states that is also very robust to suboptimal initializations.

We start the second section with a novel stick-breaking definition of a certain class of Bayesian nonparametric priors called gamma processes (GP), using its characterization as a completely random measure and attendant Poisson process machinery. We use it to derive a variational inference algorithm for the infinite Gamma-Poisson model, a particular Bayesian nonparametric latent structure formulation where the latent variables are drawn from a GP prior with Poisson likelihoods. We present results on probabilistic matrix

factorization (PMF) tasks on document corpora, and show that we compare favorably to similar constructive methods from the literature.

In the third section, we use concepts from statistical physics to develop a robust Monte Carlo sampler that efficiently traverses the parameter space. Built on the Hamiltonian Monte Carlo framework, our sampler uses a modified Nosé-Poincaré Hamiltonian with a suitable symplectic leapfrog integrator to ensure correct sampling from the canonical ensemble, with structural cues from Riemannian preconditioning matrices. We follow it up with variant that uses minibatched stochastic gradients, to handle real-life machine learning scenarios with massive datasets, showing strong performance in the PMF scenario mentioned above.

We continue with an L-BFGS optimization algorithm on Riemannian manifolds that uses stochastic variance reduction techniques for fast convergence with constant step sizes, without resorting to standard linesearch methods, and provide a new convergence proof for strongly convex functions without using curvature conditions on the manifold. Compared to state-of-the-art Euclidean methods like VR-PCA and first-order Riemannian techniques, our method performs strongly in computation of Karcher means for symmetric p.d. matrices and leading eigenvalues of large data matrices.

We finish with a novel technique for learning the mass matrices in Monte Carlo samplers obtained from discretized dynamics that preserve some energy function, by using existing dynamics in the sampling step of a Monte Carlo EM framework, and learning the mass matrices in the M step with a simple but effective online technique. Along with a novel stochastic sampler based on Nosé-Poincaré dynamics, we use this framework with standard Hamiltonian Monte Carlo as well as newer stochastic algorithms such as SGHMC and SGNHT, achieving sampling accuracies comparable to existing adaptive samplers that use Riemannian preconditioning techniques, while being significantly faster on a variety of datasets.

Dedicated to my parents

Acknowledgments

I would like to thank my advisors, Prof. Srinivasan Parthasarathy and Prof. Brian Kulis for their support, ideas, counsel and insights over the last five years; their advice and encouragement has been invaluable in times of unforeseen disquiet. I also thank my committee members, Prof. Mikhail Belkin and Prof. Huan Sun for their helpful advice and insights. I have been fortunate to have enjoyable internships in industry, with interesting projects and excellent collaborators; I thank my internship mentors for these opportunities. I thank the staff at the Department of Computer Science and Engineering here at Ohio State for playing an invaluable role in making the overall experience memorable. I would like to thank my friends here in Columbus for their enjoyable company, and for making these last few years fly by. Lastly, my heartfelt gratitude to my family for their constant love, support and encouragement over the course of my studies, and indeed my entire life.

This dissertation is based in part upon work supported by the National Science Foundation under Grant Numbers IIS-1217433, DMS-1418265 and CCF-1645599. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author (and his advisor and collaborators where applicable) and do not necessarily reflect the views of the National Science Foundation or The Ohio State University.

Vita

December 2010	B.E., Computer Science and Engineering, Jadavpur University, India.
August 2012 – July 2013	University Fellow, The Ohio State University, USA.
May 2017	M.S., Computer Science and Engineering, The Ohio State University, USA.
August 2013 – present	Graduate Research/Teaching Associate, The Ohio State University, USA.

Publications

A. Roychowdhury, K. Jiang, and B. Kulis. Small-Variance Asymptotics for Hidden Markov Models. In *Advances in Neural Information Processing Systems (NIPS)*, 2013.

A. Roychowdhury and B. Kulis. Gamma Processes, Stick-Breaking and Variational Inference. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2015.

A. Roychowdhury, B. Kulis and S. Parthasarathy. Robust Monte Carlo Sampling with Riemannian Nosé-Poincaré Hamiltonian Dynamics. In *International Conference on Machine Learning (ICML)*, 2016.

A. Roychowdhury and S. Parthasarathy. Accelerated Stochastic Quasi-Newton Optimization on Riemannian Manifolds. *In submission*, 2017.

A. Roychowdhury and S. Parthasarathy. Adaptive Bayesian Sampling with Monte Carlo EM. In *Advances in Neural Information Processing Systems (NIPS)*, 2017.

Fields of Study

Major Field: Computer Science and Engineering

Table of Contents

Page

List of Tables

Table **Page**

List of Figures

Figure **Page**

Chapter 1: Introduction

Modern machine learning leverages a variety of techniques from the statistical and mathematical literature in the quest to extract meaningful patterns from large scale data and to perform cogent inference based on them. One needs a rigorous set of tools to mathematically model diverse types of data, as well as develop fast and efficient methods to train and gather insights from these constructions. Probabilistic graphical models offer a powerful way to model various types of data, by imposing complex, often high dimensional, probability distributions on the data at hand. For small prototypical datasets this does not pose too much of a challenge, but simple parametric models often fail to capture all the nuances of very large data. A simple example with mixture models illustrates the point: one can use a mixture of Gaussian distributions to model any set of clustered datapoints, but correctly estimating the number of such clusters is a nontrivial problem and often requires strong assumptions and/or domain knowledge. The lack of either of these can lead to incorrectly specified models, rendering subsequent parameter estimates useless and providing very few actionable insights into the data.

Nonparametric techniques drawn from the statistical literature offer a robust way to deal with such issues, allowing one to create models with an unbounded number of parameters that adapt to the data. Even then, one needs to address the question of uncertainty in the model estimates, a task of arguably greater complexity in these situations as opposed to simpler parametric cases. One extensively studied way to do this uses Bayesian techniques

to bake one's background information and assumptions into "prior" probability distributions in the model, and adaptively update these using the observed data in the learning phase. This leads to a set of tools known as Bayesian nonparametrics, which have seen widespread adoption in the machine learning community in recent times.

Once an appropriate model has been selected, the key challenge lies in devising efficient methods to learn the various parameters associated with the model. This is a nontrivial problem because: a) the number of such parameters can grow exponentially with the size of the data, leading to significant challenges in applying promising prototypical techniques at scale, and b) most real-life problems are best modeled by complex nonparametric models for which closed-form learning and inference techniques are not known to exist. This therefore requires one to develop approximate techniques for parameter estimation that should scale well, while preserving as many of the theoretical guarantees of exact methods derived for simpler models as possible, in order to acquire reasonably rigorous estimates of correctness.

1.1 Thesis Statement.

We posit that scalable learning and inference algorithms are necessary for expanding the scope of Bayesian nonparametric probabilistic models to truly large-scale datasets. Doing so requires developing scalable variants of analytic closed form learning methods that preserve many of the theoretical niceties, from both algorithmic and underlying modeling perspectives. To that end, we propose a set of methods using techniques such as stochastic MCMC sampling, stochastic optimization in both Euclidean and Riemannian manifold domains, variational inference and small-variance asymptotics, that can be applied to large scale data modeled using rich Bayesian nonparametric paradigms, while scaling to large datasets.

1.2 Contributions and Organization of the Thesis.

In the rest of this thesis we discuss various novel techniques we have developed recently for improving the robustness and scalability of the learning process for Bayesian nonparametric models, and mention some interesting avenues of extending this work using novel optimization techniques. We posit the small-variance asymptotic reduction of Bayesian nonparametric hidden Markov models to significantly more scalable combinatorial optimization problems, variational inference approaches derived from a novel constructive definition of Gamma processes to the problem of nonparametric count-based topic modeling, and the suitability of stochastic Hamiltonian Monte Carlo approaches with Riemannian preconditioning to the problem of efficiently sampling from otherwise intractable high-dimensional posterior distributions. We follow that up with fast quasi-Newton optimization techniques on Riemannian manifolds, as well as another Monte Carlo sampler using a fast alternative to learning the mass matrices using the Monte Carlo EM framework.

Small-Variance Asymptotics for Hidden Markov Models. Our first approach proposes a new way of learning hidden Markov models with Bayesian nonparametric augmentations. Parametric hidden Markov models are a venerable approach to modeling sequential data, with wide applications in many fields of science. In the machine learning literature, attempts to use Bayesian nonparametric ideas in conjunction with these have led to the development of the infinite HMM, where one imposes a Dirichlet Process or hierarchical DP prior on the transition matrix of the HMM, leading to a countable infinite state space, allowing the model to dynamically learn the best number of hidden states and their transition dynamics from the data. Inference for this problem however has proven to be challenging, with the difficulties imposed by the sequential nature of the data combining with the nontrivialities of learning DP-based mixture models to impose significant scalability issues on sampling-based techniques. We attempt to mitigate this problem with the use of the recently proposed

technique of small-variance asymptotics. This method is influence by the connection between Gaussian mixture learning , in the parametric case, and K-means; in particular, if one assumes a mixture of isotropic Gaussians and takes the limit of the update steps in the EM algorithm with the common variance tending to zero, then the K-means algorithm is obtained. Alternatively, the limit of the expected log-likelihood of the EM problem gives rise to the objective function of the K-means algorithm. This idea has been applied to remarkable effect in Bayesian nonparametric situations as well, where suitable formulation of the probability distributions followed by the small-variance limit allows one to obtain hard-assignment optimization problems from sampling-based algorithms or directly from the joint log-likelihoods. We apply this technique to both parametric and Bayesian nonparametric variants of hidden Markov models to arrive at, in the former case, the Viterbi re-estimation problem, and a highly scalable approach to learning infinite HMM's in the latter.

Variational Inference for a Novel Bayesian Nonparametric Construction. We propose a new way of constructing certain Bayesian nonparametric priors, and derive a fast inference algorithm for that class. The first such constructive definition of Bayesian nonparametric priors started with Sethuraman's formulation of the Dirichlet process as an infinite sum of suitably drawn independent random variables. This construction has led to the creation of multiple learning algorithms in machine learning, since DP's are the Bayesian nonparametric prior of choice in situations ranging from modeling the distribution of the mixture weights in infinite Gaussian mixtures, forming the basis for hierarchical Dirichlet processes which are extensively used in topic modeling in textual data, to the creation of dependent DP's for use with nonparametric sequence learning. This construction was recently used to derive a definition of Beta processes, which form the basis for Indian Buffet Processes for use with learning latent feature allocations, and subsequently to the development of variational inference algorithms for IBP's. We extend this line of work by deriving a new construction

for Gamma processes, which are beginning to be used for count modeling situation, where instead of learning binary feature assignment matrices, one attempts to model the counts of features seen in observations, and builds models on top of that. Topic modeling using Poisson matrix factorization is a straightforward application, where instead of using DP's and multinomial distributions to model the latent topic distributions and word assignments, one attempts to factorize the observed counts of words in documents by fitting a Gamma process prior on the latent topic distributions and independent Poissons on the likelihoods. Till our contribution, only Monte Carlo-based algorithms had been applied to these problems, with accompanying scalability issues. Our construction of the infinite Gamma process and a variational inference algorithm provide an alternative way of learning these count matrices, showing promising performance in topic modeling problems.

A New MCMC Sampling Algorithm with Riemannian Preconditioning. Our third technique proposes a new sampling method intended to exploit the geometry of the parameter space for more efficient exploration, along with other tricks. The basic idea falls under the umbrella of Monte Carlo sampling, a set of methods designed to streamline the process of drawing samples from probability distributions. Such methods are almost indispensable in Bayesian nonparametrics, since, as mentioned earlier, the process of obtaining uncertainty estimates requires the use of complex probabilistic prior distributions, and the corresponding data-driven closed-from posteriors do not usually admit tractable closed-form estimates. Simple random walk-based methods derived as discretized versions of Wiener processes were initially used for such problems, but their inefficiencies in traversing complicated high-dimensional parameter spaces were soon exposed. This led to the development of second-order techniques using ideas from statistical physics, called Hamiltonian Monte Carlo. Till recently, such methods have been restricted to relatively small use-cases, primarily due to the problem of correctly selecting the values of certain matrices essential

to the estimation process, usually of size quadratic in the dimensionality, and the use of Metropolis-Hastings-based rejection techniques for dropping unsuitable samples. The main issue introduced by the latter is the use of the entire dataset to calculate the acceptance probabilities, thereby limiting its use in large-data settings. Our work proposes a method that addresses both of thee issues, by adaptively learning the all-important matrices mentioned above using ideas from statistical physics and Riemannian geometry, in a stochastic setting that only uses a small subset of the data at each step, and can be proved to converge correctly even without performing the expensive acceptance ratio computations.

Stochastic Quasi-Newton Optimization on Riemannian Manifolds. Optimization algorithms are a rich area of study in machine learning, and crop up in almost all learning and inference problems. The variational inference techniques, for instance, are at their heart optimization problems where one minimizes KL-divergences between probability distributions. Application of small-variance asymptotics, as mentioned above, also gives rise to combinatorial optimization problems. Therefore, solving such problems quickly is a necessary step to attain desired scalability and efficiency targets. To this end, we aim to investigate the use of recent approaches to solving constrained optimization problems by casting them on suitable Riemann manifolds, and their applications in the algorithms mentioned above. A simple example would be learning the leading eigenvectors in a PCA setting, where one has unit-norm constraints on the solution vectors. This constraint restricts the vectors to lie on a sphere, the surface of which is known to be a well-behaved Riemann manifold. The problem can therefore be transformed into an unconstrained optimization on the manifold, and indeed it has been shown that corresponding "Riemannian optimization" algorithms rival and sometimes outperform state-of-the-art Euclidean optimizers for such tasks. Another interesting example is the recent application of quasi-Newton methods (L-BFGS) to the task of learning Gaussian mixtures, where a simple reparametrization leads

to the casting of the nonconvex maximum likelihood problem into a convex optimization problem on a product of Riemann manifolds, with the geodesics playing the role of distance minimizing curves. Our contribution is along these lines; we propose a novel L-BFGS algorithm on Riemannian manifolds that uses stochastic variance reduction techniques for responsible convergence in a big-data setting, for both convex and nonconvex problems on manifolds. We provide a novel proof of convergence for the convex case, without directly porting the proofs of similar algorithms in the Euclidean domain. We also show strong performance for the algorithm in synthetic experiments on computing the Karcher means of positive definite matrices and learning the leading eigenvectors of large data matrices, two problems that are quite common in machine learning applications, converging noticeably faster than a state-of-the-art Euclidean algorithm as well as first-order Riemannian methods with stochastic variance reduction augmentations.

Using Monte Carlo EM for Adaptive Sampling. Our final contribution attacks the problem of learning the "mass" matrices in certain Monte Carlo sampling algorithms. As alluded to earlier, MCMC samplers form an integral part of the machine learning practitioner's toolkit, being the only way to draw samples of parameters from complex probability distributions in a provably correct way. Hamiltonian Monte Carlo methods are a popular class of methods in this general area, but tuning these optimally is a major challenge. Some of the issues one has to be concerned with in these cases are: discretizing the differential equations arising from these systems correctly, possibly applying correction terms to handle stochastic noise arising minibatched gradients, selecting the step sizes and number of "leapfrog" steps in the discretized system, and finally selecting correct values for the mass matrix. While the stepsizes and leapfrog steps can be optimized via simple grid search methods, selecting correct values for the mass is difficult, since it scales as the square of the dimensionality of the parameters, thereby easily running into the thousands for moderately

sized datasets. Other than setting these to the identity matrix, the primary way of dealing with these parameters is via the Riemannian manifold formulation mentioned above, where one formulates these matrices in terms of the parameters to be sampled, and implicitly "learns" them as the sampler runs. While effective, this method often leads to significant reformulations of the resulting samplers, as we describe later, and imposes heavy penalties on the runtimes. This has led to such methods being restricted to low dimensional datasets, where the tradeoff between the improved sampling efficiency and greatly increased runtimes is tolerable. We circumvent this problem by proposing a completely new way of learning these mass matrices, using the Monte Carlo EM framework from the statistics literature. As we describe later, we can wrap any existing sampler in this framework, keeping the energy function formulation and sampling dynamics unchanged. Put simply, we cast the problem of learning the mass matrix into a maximum likelihood problem of learning the precision of a certain Gaussian distribution that encodes the "kinetic energy" of such systems, using samples collected from running the existing sampler as observations. We use this method to develop adaptive variants of a number of existing samplers, without the expensive overhead of a Riemannian reformulation. The resulting algorithms perform close to or better than Riemannian samplers, while being an order of magnitude faster and notably more simpler to implement and optimize.

The rest of this thesis is organized in the order mentioned above: Chapter 2 discusses our inference algorithms for hidden Markov models, both standard and nonparametric, using small-variance asymptotics. In Chapter 3 we describe our novel construction of the gamma process using Poisson process machinery and "stick-breaking" ideas. Chapter 4 contains our Monte Carlo sampling algorithms derived from the Nosé-Poincaré Hamiltonian with Riemannian preconditioning. In Chapter 5 we discuss our stochastic L-BFGS algorithm for Riemannian manifolds, and provide convergence proofs and synthetic experiments.

Chapter 6 contains our second set of Monte Carlo samplers, this time using the Monte Carlo EM framework to adaptively learn the mass matrices, as an alternative to the Riemannian preconditioning methods discussed in Chapter 4. The material in these chapters is composed of work published in [RJK13, RK15, RKP16, RP17a] and [RP17b]. We conclude this dissertation in Chapter 7 with a summary and discussion of future directions.

Chapter 2: Small-Variance Asymptotics for Hidden Markov Models

Large scale probabilistic models applied to modern "big-data" problems provide a unique set of challenges for inference. Graphical models being the undisputed choice for building rich probability distributions, scaling up existing sampling methods and variational inference techniques is not feasible for certain classes of applications.

A recent line of work along these lines has been based on *small-variance asymptotics* (SVA), which performs asymptotics on certain formulations of latent variables in probabilistic graphical models, providing a mechanism of deriving scalable combinatorial optimization algorithms from these models, while preserving their structural benefits. A canonical motivation would be the connection between Gaussian mixture models and K-Means: in the limit as the common variance of the (spherical) Gaussian distributions goes to zero, the maximum likelihood problem for this graphical models turns into optimization of the K-Means objective function; the *EM* method for the mixture model can be seen to approach Lloyd's algorithm as well. This basic idea has recently been applied to models beyond Gaussian mixtures—to Dirichlet process mixtures and hierarchical Dirichlet processes [KJ12], to Bayesian nonparametric mixtures of exponential family observations [JKJ12], and learning latent features using the Beta process [BKJ13]. SVA allows one to derive straightforward algorithms for these models, that are significantly more scalable than samplers or variational inference approaches while preserving many of the benefits of the probabilistic models.

Prior to this work, SVA had been applied only to Bayesian mixture models, both parametric and nonparametric, without any temporal properties. To the best of our knowledge, there was no existing work applying these methods to Hidden Markov models (HMM) or its Bayesian nonparametric counterpart, the infinite Hidden Markov model (iHMM). HMMs are probably the most widely used graphical model for discrete sequence data, with applications including DNA sequence analysis, natural language processing and speech recognition [Bis06]. These models assume that the observations are independently generated from discrete hidden states, which evolve according to Markov assumptions. The goal of the learning algorithm is to estimate the transition and emission distributions given only the observed data.

We begin our discussion of scalable algorithms for sequential probabilistic models by applying SVA to the standard parametric HMM. Following a certain parametrization of the latent variables, we get a penalized K-Means objective function in the limit as the variance goes to zero, with the penalties being a function of the state transitions. As a special case of this objective, we obtain the segmental K-Means of [Rab89]. For the Bayesian nonparametric case, the iHMM, we obtain an objective that combines the asymptotics of the parametric case with those of the Hierarchical Dirichlet Process from [KJ12]; in particular, we derive a penalized K-Means objective with three penalties: one for state transitions, one for the number of states reachable from each state, and one for the total number of states. This allows us to use dynamic programming to optimize the objective, something one cannot do for the standard sampler for the infinite HMM; we describe a simple algorithm that monotonically decreases the objective function. We compare our non-probabilistic "hard" algorithms to their probabilistic counterparts, using experiments on a number o real and synthetic datasets.

2.1 Related Work.

There are several algorithms for maximum likelihood estimation for the standard parametric HMM, such as the Baum-Welch algorithm (a special instance of EM), and the segmental K-Means algorithm of [Rab89]. HMMs with an unbounded number of latent states, known as infinite HMMs [BGR02, TJBB06] are Bayesian nonparametric extensions of the parametric models with hierarchical Dirichlet process priors on the transition probabilities, allowing a countably infinite state space. Exact inference in these models is intractable, so one typically uses Markov chain Monte Carlo-based sampling methods. However Gibbs sampling is known to be extremely slow to converge for this model [TJBB06], and is unable to exploit the forward-backward structure of HMMs. [GSTG08] presents a method which bypasses this obstacle via slice sampling, where only a finite number of hidden states are considered in each iteration. Since it works in the non-collapsed space, it is still computationally expensive. Thus infinite-state HMMs, while desirable from a modeling perspective, have been limited by their inability to scale to large data sets—this is precisely the situation in which small-variance asymptotics has the potential to be beneficial.

Beyond the connection between Gaussian mixtures and K-Means discussed earlier, we should note that similar connections between probabilistic PCA and standard PCA have been studied in [TB99, Row98], while those between support vector machines and a restricted Bayes optimal classifier have been discussed in [TK00].

2.2 Asymptotics of the finite-state HMM

As a warm-up, we begin with the simpler parametric (finite-state) hidden Markov model, and show that small-variance asymptotics on the joint log-likelihood yields a penalized k-means objective, and on the EM algorithm yields a generalized segmental k-means algorithm.

The tools developed in this section will then be used for the more involved nonparametric model.

2.2.1 The Model

The Hidden Markov Model assumes a hidden state sequence $\mathscr{Z} = \{z_1, \ldots, z_N\}$ drawn from a finite discrete state space $\{1, \ldots, K\}$, coupled with the observation sequence $\mathscr{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$. The resulting generative model defines a probability distribution over the hidden state sequence $\mathscr{Z}$ and the observation sequence $\mathscr{X}$. Let $T \in \mathbb{R}^{K \times K}$ be the stationary transition probability matrix of the hidden state sequence with $T_{i\cdot} = \boldsymbol{\pi}_i \in \mathbb{R}^K$ being a distribution over the latent states. For clarity of presentation, we will use a binary 1-of-K representation for the latent state assignments. That is, we will write the event of the latent state at time step t being k as $z_{tk} = 1$ and $z_{tl} = 0 \quad \forall l = 1...K, l \neq k$. Then the transition probabilities can be written as $T_{ij} = \Pr(z_{tj} = 1 | z_{t-1,i} = 1)$. The initial state distribution is $\boldsymbol{\pi}_0 \in \mathbb{R}^K$. The Markov structure dictates that $z_t \sim \text{Mult}(\boldsymbol{\pi}_{z_{t-1}})$, and the observations follow $\boldsymbol{x}_t \sim \Phi(\theta_{z_t})$. The observation density Φ is assumed invariant, and the Markov structure induces conditional independence of the observations given the latent states.

In the following, we present the asymptotic treament for the finite HMM with Gaussian emission densities $\Pr(\boldsymbol{x}_t | z_{tk} = 1) = \mathscr{N}(\boldsymbol{x}_t | \boldsymbol{\mu}_k, \sigma^2 I_d)$. Here $\theta_{z_t} = \boldsymbol{\mu}_k$, since the parameter space Θ contains only the emission means. Generalization to exponential family emission densities is straightforward[JKJ12]. At a high level, the connection we seek to establish can be proven in two ways. The first approach is to examine small-variance asymptotics directly on the the joint probability distribution of the HMM, as done in [BKJ13] for clustering and feature learning problems. We will primarily focus on this approach, since our ideas can be more clearly expressed by this technique, and it is independent of any inference algorithm. The other approach is to analyze the behavior of the EM algorithm as the variance goes

to zero. We will briefly discuss this approach as well, but for further details the interested reader can consult the chapter appendix.

2.2.1.1 Exponential Family Transitions

Our main analysis relies on appropriate manipulation of the transition probabilities, where we will use the bijection between exponential families and Bregman divergences established in [BMDG05]. Since the conditional distribution of the latent state at any time step is multinomial in the transition probabilities from the previous state, we use the aforementioned bijection to refactor the transition probabilities in the joint distribution of the HMM into a form than utilizes Bregman divergences. This, with an appropriate scaling to enable small-variance asymptotics as mentioned in [JKJ12], allows us to combine the emission and transition distributions into a simple objective function.

We denote $T_{jk} = \Pr(z_{tk} = 1 | z_{t-1,j} = 1)$ as before, and the multinomial distribution for the latent state at time step t as

$$\Pr(\boldsymbol{z}_t | z_{t-1,j} = 1) = \prod_{k=1}^{K} T_{jk}^{z_{tk}}. \tag{2.1}$$

In order to apply small-variance asymptotics, we must allow the variance in the transition probabilities to go to zero in a reasonable way. Following the treatment in [BMDG05], we can rewrite this distribution in a suitable exponential family notation, which we then express in the following equivalent form:

$$\Pr(\boldsymbol{z}_t | z_{t-1,j} = 1) = \exp(-d_\phi(\boldsymbol{z}_t, \boldsymbol{m}_j)) b_\phi(\boldsymbol{z}_t), \tag{2.2}$$

where the Bregman divergence $d_\phi(\boldsymbol{z}_t, \boldsymbol{m}_j) = \sum_{k=1}^{K} z_{tk} \log(z_{tk}/T_{jk}) = \mathrm{KL}(\boldsymbol{z}_t, \boldsymbol{m}_j)$, $\boldsymbol{m}_j = \{T_{jk}\}_{k=1}^{K}$ and $b_\phi(\boldsymbol{z}_t) = 1$. See the notes in the appendix for derivation details. The prime motivation for using this form is that we can appropriately scale the variance of the exponential family distribution following Lemma 3.1 of [JKJ12]. In particular, if we introduce a new parameter

$\hat{\beta}$, and generalize the transition probabilities to be

$$\Pr(\boldsymbol{z}_t|z_{t-1,j}=1) = \exp(-\hat{\beta}d_\phi(\boldsymbol{z}_t,\boldsymbol{m}_j))b_{\tilde{\phi}}(\boldsymbol{z}_t),$$

where $\tilde{\phi} = \hat{\beta}\phi$, then the mean of the distribution is the same in the scaled distribution (namely, $\boldsymbol{m}_j$) while the variance is scaled. As $\hat{\beta} \to \infty$, the variance goes to zero.

The next step is to link the emission and transition probabilities so that the variance is scaled appropriately in both. In particular, we will define $\beta = 1/2\sigma^2$ and then let $\hat{\beta} = \lambda\beta$ for some λ. The Gaussian emission densities can now be written as $\Pr(\boldsymbol{x}_t|z_{tk}=1) = \exp(-\beta\|\boldsymbol{x}_t - \boldsymbol{\mu}_k\|_2^2)f(\beta)$ and the transition probabilities as $\Pr(\boldsymbol{z}_t|z_{t-1,j}=1) = \exp(-\lambda\beta d_\phi(\boldsymbol{z}_t,\boldsymbol{m}_j))b_{\tilde{\phi}}(\boldsymbol{z}_t)$. See [JKJ12] for further details about the scaling operation.

2.2.1.2 Joint Probability Asymptotics

We now have all the background development required to perform small-variance asymptotics on the HMM joint probability, and derive the segmental k-means algorithm. Our parameters of interest are the $Z = [\boldsymbol{z}_1,...,\boldsymbol{z}_N]$ vectors, the $\boldsymbol{\mu} = [\boldsymbol{\mu}_1,...,\boldsymbol{\mu}_K]$ means, and the transition parameter matrix T. We can write down the joint likelihood by taking a product of all the probabilities in the model:

$$p(\mathscr{X},\mathscr{Z}) = p(\boldsymbol{z}_1)\prod_{t=2}^{N} p(\boldsymbol{z}_t|\boldsymbol{z}_{t-1})\prod_{t=1}^{N}\mathscr{N}(\boldsymbol{x}_t|\boldsymbol{\mu}_{\boldsymbol{z}_t},\sigma^2 I_d),$$

With some abuse of notation, let $\boldsymbol{m}_{\boldsymbol{z}_{t-1}}$ denote the mean transition vector given by the assignment $\boldsymbol{z}_{t-1}$ (that is, if $z_{t-1,j}=1$ then $\boldsymbol{m}_{\boldsymbol{z}_{t-1}} = \boldsymbol{m}_j$). The exp-family probabilities above allow us to rewrite this joint likelihood as

$$p(\mathscr{X},\mathscr{Z}) \propto \exp\left[-\beta\left(\sum_{t=1}^{N}\|\boldsymbol{x}_t - \boldsymbol{\mu}_{\boldsymbol{z}_t}\|_2^2 + \lambda\sum_{t=2}^{N}\mathrm{KL}(\boldsymbol{z}_t,\boldsymbol{m}_{\boldsymbol{z}_{t-1}})\right) + \log p(\boldsymbol{z}_1)\right]. \tag{2.3}$$

To obtain the corresponding non-probabilistic objective from small-variance asymptotics, we consider the MAP estimate obtained by maximizing the joint likelihood with respect to

the parameters asymptotically as σ^2 goes to zero (β goes to ∞). In our case, it is particularly simple given the joint likelihood above. The log-likelihood easily yields the following asymptotically:

$$\max_{Z,\boldsymbol{\mu},T} -\left(\sum_{t=1}^{N} \|\boldsymbol{x}_t - \boldsymbol{\mu}_{z_t}\|_2^2 + \lambda \sum_{t=2}^{N} \mathrm{KL}(\boldsymbol{z}_t, \boldsymbol{m}_{\boldsymbol{z}_{t-1}})\right) \tag{2.4}$$

or equivalently,

$$\min_{Z,\boldsymbol{\mu},T} \left(\sum_{t=1}^{N} \|\boldsymbol{x}_t - \boldsymbol{\mu}_{z_t}\|_2^2 + \lambda \sum_{t=2}^{N} \mathrm{KL}(\boldsymbol{z}_t, \boldsymbol{m}_{\boldsymbol{z}_{t-1}})\right). \tag{2.5}$$

Note that, as mentioned above, $\boldsymbol{m}_j = \left\{T_{jk}\right\}_{k=1}^{K}$. We can view the above objective function as a "penalized" k-means problem, where the penalties are given by the transitions from state to state.

One possible strategy to minimize (2.5) would be to iteratively minimize with respect to each of the individual parameters $(Z, \boldsymbol{\mu}, T)$ keeping the other two fixed. When fixing $\boldsymbol{\mu}$ and T, and taking $\lambda = 1$, the solution for Z in (2.4) is identical to the MAP update on the latent variables Z for this model, as in a standard HMM. When $\lambda \neq 1$, a simple generalization of the standard forward-backward routine can be used to find the optimal assignments. Keeping Z and T fixed, the update on $\boldsymbol{\mu}_k$ is easily seen to be the equiweighted average of the data points which have been assigned to latent state k in the MAP estimate (it is the same minimization as in k-means for updating cluster means). Finally, since $\mathrm{KL}(\boldsymbol{z}_t, \boldsymbol{m}_j) \propto -\sum_{k=1}^{K} \log T_{jk}$, minimization with respect to T simply yields the empirical transition probabilities, that is $T_{jk,new} = \frac{\text{\# of transitions from state } j \text{ to } k}{\text{\# of transitions from state } j}$, both counts from the MAP path computed during maximization w.r.t Z. We observe that, when $\lambda = 1$, the iterative minimization algorithm to solve (2.5) is exactly the segmental k-means algorithm, also known as Viterbi re-estimation.

2.2.1.3 EM algorithm asymptotics

We can reach the same algorithm alternatively by writing down the steps of the EM algorithm and exploring the small-variance limit of these steps, analogous to the approach of [KJ12] for a Dirichlet process mixture. Given space limitations (and the fact that the resulting algorithm is identical, as expected), a more detailed discussion can be found in the appendix for this chapter.

2.3 Asymptotics of the Infinite Hidden Markov Model

We now tackle the more complex nonparametric model. We will derive the objective function directly as in the parametric case but, unlike the parametric version, we will not apply asymptotics to the existing sampler algorithms. Instead, we will present a new algorithm to optimize our derived objective function. By deriving an algorithm directly, we ensure that our method takes advantage of dynamic programming, unlike the standard sampler.

2.3.1 The Model

The iHMM, also known as the HDP-HMM [BGR02, TJBB06] is a nonparametric Bayesian extension to the HMM, where an HDP prior is used to allow for an unspecified number of states. The HDP is a set of Dirichlet Processes (DPs) with a shared base distribution, that are themselves drawn from a Dirichlet process [TJBB06]. Formally, we can write $G_k \sim \mathrm{DP}(\alpha, G_0)$ with a shared base distribution $G_0 \sim \mathrm{DP}(\gamma, H)$, where H is the global base distribution that permits sharing of probability mass across G_ks. α and γ are the concentration parameters for the G_k and G_0 measures, respectively.

To apply HDPs to sequential data, the iHMM can be formulated as follows:

$$\beta \sim \mathrm{GEM}(\gamma), \quad \boldsymbol{\pi}_k|\beta \sim \mathrm{DP}(\alpha, \beta), \quad \theta_k \sim H,$$

$$\boldsymbol{z}_t|\boldsymbol{z}_{t-1} \sim \text{Mult}(\boldsymbol{\pi}_{\boldsymbol{z}_{t-1}}), \qquad \boldsymbol{x}_t \sim \Phi(\theta_{\boldsymbol{z}_t}).$$

For a full Bayesian treatment, Gamma priors are placed on the concentration parameters (though we will not employ such priors in our asymptotic analysis).

Following the discussion in the parametric case, our goal is to write down the full joint likelihood in the above model. As discussed in [TJBB06], the Hierarchical Dirichlet Process yields assignments that follow the Chinese Restaurant Franchise (CRF), and thus the iHMM model additionally incorporates a term in the joint likelihood involving the prior probability of a set of state assignments arising from the CRF. Suppose an assignment of observations to states has K different states (i.e., number of restaurants in the franchise), s_i is the number of states that can be reached from state i in one step (i.e., number of tables in each restaurant i), and n_i is the number observations in each state i (i.e., number of customers in each restaurant). Then the probability of an assignment in the HDP can be written as (after integrating out mixture weights [Ant74, Sud12], and if we only consider terms that would not be constants after we do the asymptotic analysis [BKJ13]):

$$p(\mathscr{Z}|\alpha,\gamma,\lambda) \propto \gamma^{K-1} \frac{\Gamma(\gamma+1)}{\Gamma(\gamma+\sum_{k=1}^{K} s_k)} \prod_{k=1}^{K} \alpha^{s_k-1} \frac{\Gamma(\alpha+1)}{\Gamma(\alpha+n_i)}.$$

For the likelihood, we follow the same assumption as in the parametric case: the observation densities are Gaussians with a shared covariance matrix $\sigma^2 I_d$. Further, the means are drawn independently from the prior $\mathscr{N}(0,\rho^2 I_d)$, where $\rho^2 > 0$ (this is needed, as the model is fully Bayesian now). Therefore, $p(\boldsymbol{\mu}_{1:K}) = \prod_{k=1}^{K} \mathscr{N}(\boldsymbol{\mu}_k|0,\rho^2 I_d)$, and

$$p(\mathscr{X},\mathscr{Z}) \propto p(\mathscr{Z}|\alpha,\gamma,\lambda) \cdot p(\boldsymbol{z}_1) \prod_{t=2}^{N} p(\boldsymbol{z}_t|\boldsymbol{z}_{t-1}) \cdot \prod_{t=1}^{N} \mathscr{N}(\boldsymbol{x}_t|\boldsymbol{\mu}_{\boldsymbol{z}_t},\sigma^2 I_d) \cdot p(\boldsymbol{\mu}_{1:K}).$$

Now, we can perform the small-variance analysis on the generative model. In order to retain the impact of the hyperparameters α and γ in the asymptotics, we can choose some constants $\lambda_1,\lambda_2 > 0$ such that

$$\alpha = \exp(-\lambda_1\beta),\ \gamma = \exp(-\lambda_2\beta),$$

where $\beta = 1/(2\sigma^2)$ as before. Note that, in this way, we have $\alpha \to 0$ and $\gamma \to 0$ when $\beta \to \infty$.

We now can consider the objective function for maximizing the generative probability as we let $\beta \to \infty$. This gives $p(\mathscr{X}, \mathscr{Z}) \propto$

$$\exp\Big[-\beta\left(\sum_{t=1}^{N}\|\boldsymbol{x}_t - \boldsymbol{\mu}_{z_t}\|^2 + \lambda\sum_{t=2}^{N}\mathrm{KL}(\boldsymbol{z}_t, \boldsymbol{m}_{\boldsymbol{z}_{t-1}}) + \lambda_1\sum_{k=1}^{K}(s_k - 1) + \lambda_2(K-1)\right) + \log(p(\boldsymbol{z}_1))\Big]. \tag{2.6}$$

Therefore, maximizing the generative probability is asymptotically equivalent to the following optimization problem:

$$\min_{K,Z,\boldsymbol{\mu},T} \quad \sum_{t=1}^{N}\|\boldsymbol{x}_t - \boldsymbol{\mu}_{z_t}\|^2 + \lambda\sum_{t=2}^{N}\mathrm{KL}(\boldsymbol{z}_t, \boldsymbol{m}_{\boldsymbol{z}_{t-1}}) + \lambda_1\sum_{k=1}^{K}(s_k - 1) + \lambda_2(K-1). \tag{2.7}$$

In words, this objective seeks to minimize a penalized k-means objective, with three penalties. The first is the same as in the parametric case—a penalty based on the transitions from state to state. The second penalty penalizes the number of transitions out of each state, and the third penalty penalizes the total number of states. Note this is similar to the objective function derived in [KJ12] for the HDP, but here there is no dependence on any particular samplers. This can also be considered as MAP estimation of the parameters, since $p(\mathscr{Z}, \boldsymbol{\mu}|\mathscr{X}) \propto p(\mathscr{X}|\mathscr{Z})p(\mathscr{Z})p(\boldsymbol{\mu})$.

2.3.2 Algorithm

The algorithm presented in [KJ12] could be almost directly applied to (2.7) but it neglects the sequential characteristics of the model. Instead, we present a new algorithm to directly optimize (2.7). We follow the alternating minimization framework as in the parametric case, with some slight tweaks. Specifically, given observations $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}, \lambda, \lambda_1, \lambda_2$, our high-level algorithm proceeds as follows:

1. Initialization: initialize with one hidden state. The parameters are therefore $K = 1, \boldsymbol{\mu}_1 = \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{x}_i, T = 1$.

2. Perform a forward-backward step (via approximate dynamic programming) to update Z.

3. Update $K, \boldsymbol{\mu}, T$.

4. For each state i, ($i = 1, \ldots, K$), check if the set of observations to any state j that are reached by transitioning out of i can form a new dedicated hidden state and lower the objective function in the process. If there are several such moves, choose the one with the maximum improvement in objective function.

5. Update $K, \boldsymbol{\mu}, T$.

6. Iterate steps (2)-(5) until convergence.

There are two key changes to the algorithm beyond the standard parametric case. In the forward-backward routine (step 2), we compute the usual $K \times N$ matrix α, where $\alpha(c,t)$ represents the minimum cost over paths of length t from the beginning of the sequence and that reach state c at time step t. We use the term “cost" to refer to the sum of the distances of points to state means, as well as the additive penalties incurred. However, to see why it is difficult to compute the exact value of α in the nonparametric case, suppose we have computed the minimum cost of paths up to step $t-1$ and we would like to compute the values of α for step t. The cost of a path that ends in state c is obtained by examining, for all states i, the cost of a path that ends at i at step $t-1$ and then transitions to state c at step t. Thus, we must consider the transition from i to c. If there are existing transitions from state i to state c, then we proceed as in a standard forward-backward algorithm. However, we are also interested in two other cases—one where there are no existing transitions from i to c but

we consider this transition along with a penalty λ_1, and another where an entirely new state is formed and we pay a penalty λ_2. In the first case, the standard forward-backward routine faces an immediate problem, since when we try to compute the cost of the path given by $\alpha(c,t)$, the cost will be infinite as there is a $-\log(0)$ term from the transition probability. We must therefore alter the forward-backward routine, or there will never be new states created nor transitions to an existing state which previously had no transitions. The main idea is to derive and use bounds on how much the transition matrix can change under the above scenarios. As long as we can show that the values we obtain for α are upper bounds, then we can show that the objective function will decrease after the forward-backward routine, as the existing sequence of states is also a valid path (with no new incurred penalties).

The second change (step 4) is that we adopt a "local move" analogous to that described for the hard HDP in [KJ12]. This step determines if the objective will decrease if we create a new global state in a certain fashion; in particular, for each existing state j, we compute the change in objective that occurs when data points that transition from j to some state k are given their own new global state. By construction this step decreases the objective.

Due to space constraints, full details of the algorithm, along with a local convergence proof, are provided in the chapter appendix, §B.

2.4 Experiments

We conclude with a brief set of experiments designed to highlight the benefits of our approach. In particular, our results show that the algorithms discussed above provide clear advantages over existing parametric and non-parametric HMM algorithms in terms of speed and accuracy.

Synthetic Data. First we compare our nonparametric algorithm with the Beam Sampler for the iHMM[1]. A sequence of length 3000 was generated over a varying number of hidden states with the all-zeros transition matrix except that $T_{i,i+1} = 0.8$ and $T_{i,i+2} = 0.2$ (when $i+1 > K$, the total number of states, we choose $j = i+1 \bmod K$ and let $T_{i,j} = 0.8$, and similarly for $i+2$). Observations were sampled from symmetric Gaussian distributions with means of $\{3, 6, \ldots, 3K\}$ and a variance of 0.9.

We ran our nonparametric algorithm, the asymp-iHMM and the Beam sampler on this data, normalizing the sequence to have zero mean. We selected parameter values for the former using grid search over all three penalty terms, setting the values using a heuristic described in the appendix. For the Beam sampling algorithm, we used the following hyperparameter settings: gamma hyperpriors $(4,1)$ for α, $(3,6)$ for γ, and a zero mean normal distribution for the base H with the variance equal to 10% of the empirical variance of the dataset. The number of selected samples was varied among 10, 100, and 1000 for different numbers of states, with 5 iterations between two samples. (Note: there are no burn-in iterations and all samplers are initialized with a randomly initialized 20-state labeling.)

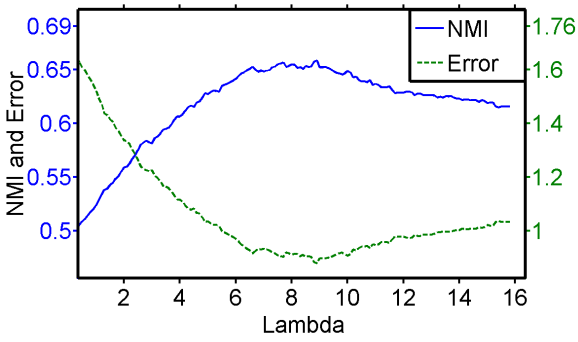


Figure 2.1: NMI and prediction error as a function of the compensation parameter λ

We show the training accuracies and running times for the two algorithms in Figure 2.2 (best viewed in color). The accuracy of the Beam sampler is given by the highest among all the samples selected. The accuracy is shown in terms of the *normalized mutual information*

[1] http://mloss.org/software/view/205/

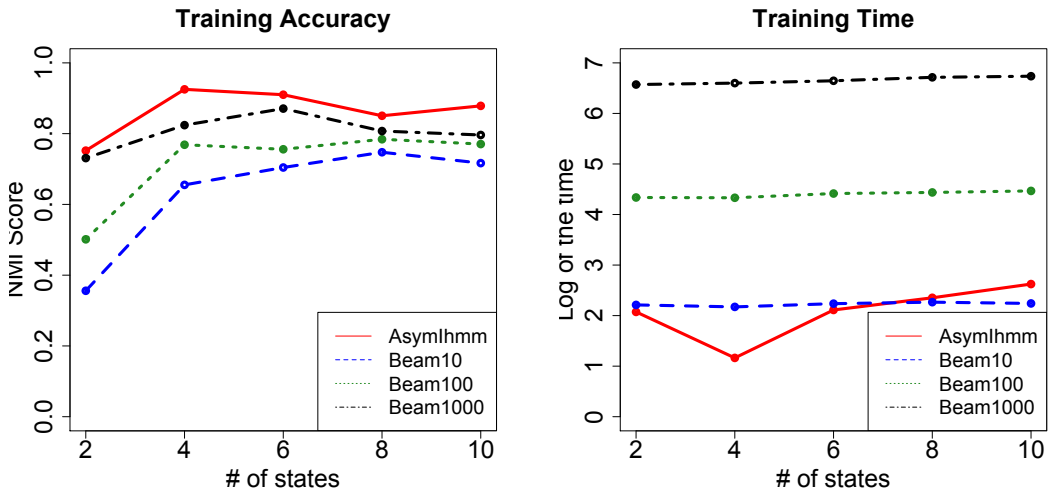


Figure 2.2: Our algorithm (asymp-iHMM) vs. the Beam Sampler on the synthetic Gaussian hidden Markov model data. (Left) The training accuracy; (Right) The training time on a log-scale.

(NMI) score (in the range of [0,1]), since the sampler may output different number of states than the ground truth and NMI can handle this situation. We can see that, in all datasets, the asymp-iHMM algorithm performs better than the sampling method in terms of accuracy, but with running time similar to the sampler with only 10 samples. For these datasets, we also observe that the EM algorithm for the standard HMM (not reported in the figure) can easily output a smaller number of states than the ground truth, which yields a smaller NMI score. We also observed that the Beam sampler is highly sensitive to the initialization of hyperparameters. Putting flat priors over the hyperparameters can ameliorate the situation, but also substantially increases the number of samples required.

Next we demonstrate the effect of the compensation parameter λ in the parametric asymptotic model, along with comparisons to the standard HMM. We will call the generalized segmental k-means of Section 2.2 the 'asymp-HMM' algorithm, shortened to 'AHMM' as appropriate. For this experiment, we used univariate Gaussians with means at 3, 6, and

10, and standard deviation of 2.9. In our ground-truth transition kernel, state i had an 80% prob. of transitioning to state $i+1$, and 10% prob. of transitioning to each of the other states. 5000 datapoints were generated from this model. The first 40% of the data was used for training, and the remaining 60% for prediction. The means in both the standard HMM and the asymp-HMM were initialized by the centroids learned by k-means from the training data. The transition kernels were initialized randomly. Each algorithm was run 50 times; the averaged results are shown in Figure 2.1.

Figure 2.1 shows the effect of λ on accuracy as measured by NMI and scaled prediction error. We see the expected tradeoff: for small λ, the problem essentially reduces to standard k-means, whereas for large λ the observations are essentially ignored. For $\lambda = 1$, corresponding to standard segmental k-means, we obtain results similar to the standard HMM, which obtains an NMI of 0.57 and error of 1.16. Thus, the parametric method offers some added flexibility via the new λ parameter.

Financial time-series prediction. Our next experiment illustrates the advantages of our algorithms in a financial prediction problem. The sequence consists of 3668 values of the Standard & Poor's 500 index on consecutive trading days from Jan 02, 1998 to July 30, 2012[2]. The index exhibited appreciable variability in this period, with both bull and bear runs. The goal here was to predict the index value on a test sequence of trading days, and compare the accuracies and runtimes of the algorithms.

To prevent overfitting, we used a training window of length 500. This window size was empirically chosen to provide a balance between prediction accuracy and runtime. The algorithms were trained on the sequence from index i to $i+499$, and then the $i+500$-th value was predicted and compared with the actual recorded value at that point in the sequence. i ranged from 1 to 3168. As before, the mixture means were initialized with k-means and the

[2]http://research.stlouisfed.org/fred2/series/SP500/downloaddata

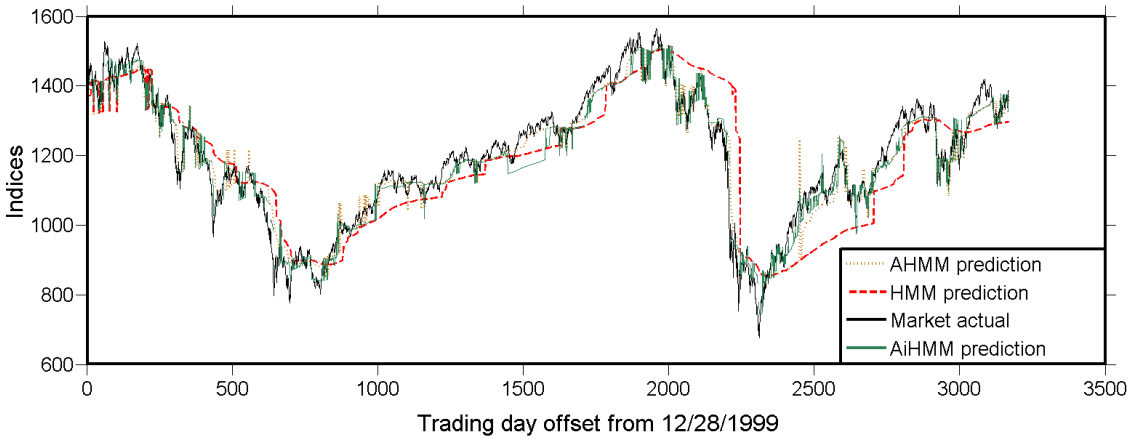


Figure 2.3: Predicted values of the S&P 500 index from 12/29/1999 to 07/30/2012 returned by the asymp-HMM, asymp-iHMM and the standard HMM algorithms, with the true index values for that period (better in color); see text for details.

transition kernels were given random initial values. For the asymp-HMM and the standard HMM, the number of latent states was empirically chosen to be 5. For the asymp-iHMM, we tune the parameters to get also 5 states on average. For predicting observation $T+1$ given observations up to step T, we used the weighted average of the learned state means, weighted by the transition probabilities given by the state of the observation at time T.

We ran the standard HMM along with both the parametric and non-parametric asymptotic algorithms on this data (the Beam sampler was too slow to run over this data, as each individual prediction took on the order of minutes). The values predicted from time step 501 to 3668 are plotted with the true index values in that time range in Figure 2.3. Both the parametric and non-parametric asymptotic algorithms perform noticably better than the standard HMM; they are able to better approximate the actual curve across all kinds of temporal fluctuations. Indeed, the difference is most stark in the areas of high-frequency oscillations. While the standard HMM returns an averaged-out prediction, our algorithms

latch onto the underlying behavior almost immediately and return noticably more accurate predictions. Prediction accuracy has been measured using the mean absolute percentage (MAP) error, which is the mean of the absolute differences of the predicted and true values expressed as percentages of the true values. The MAP error for the HMM was 6.44%, that for the asymp-HMM was 3.16%, and that for the asymp-iHMM was 2.78%. This confirms our visual perception of the asymp-iHMM algorithm returning the best-fitted prediction in Figure 2.3.

Additional Real-World Results. We also compared our methods on a well-log data set that was used for testing the Beam sampler. Due to space constraints, further discussion of these results is included in the appendix for this chapter.

2.5 Conclusion

In this chapter we discussed an asymptotic treatment of the HMM and iHMM, with the goal of obtaining nonprobabilistic formulations from the hidden Markov model and friends, thereby expanding the application of small-variance asymptotics to sequential probabilistic models. The main contribution of this section would be the dynamic programming-based algorithm for sequential models with a latent set of states of dynamic cardinality, derived by applying SVA to the infinite HMM.

2.6 Proof Details and Experimental Addenda

2.6.1 Parametric HMM Details

In this section we describe the derivation of equation (2.2) in section 2.2.1.1 in Chapter 2, and also discuss the asymptotic treatment of the EM algorithm for the finite-state HMM.

2.6.2 Bregman Divergence Transition Representation

First we recall the required equations from section 2.1.1. One can write the multinomial distribution for the latent state at time t as

$$\Pr(\boldsymbol{z}_t|z_{t-1,j}=1)=\prod_{k=1}^{K}T_{jk}^{z_{tk}}.$$

After some algebra this can be re-written as

$$\exp\left[\sum_{k=1}^{K}\log T_{jk}^{z_{tk}}\right]=\exp\left[\sum_{k=1}^{K-1}z_{tk}\cdot\log T_{jk}-z_{tK}\cdot\log\frac{1}{T_{jK}}\right].$$

Recall that, in our binary 1-of-K notation, $z_{tK}=1-\sum_{k=1}^{K-1}z_{tk}$. This allows us to write the equation above as

$$\exp\left[\sum_{k=1}^{K}\log T_{jk}^{z_{tk}}\right]=\exp\left[\sum_{k=1}^{K-1}z_{tk}\cdot\frac{\log T_{jk}}{\log T_{jK}}-\log\frac{1}{T_{jK}}\right]. \tag{2.8}$$

To reparameterize this in exponential family notation, we denote $\boldsymbol{\theta}_j^{-K}=\{\log(T_{jk}/T_{jK})\}_{k=1}^{K-1}$. Then the summation in (2.8) is clearly the inner product of $\boldsymbol{z}_t^{-K}$and $\boldsymbol{\theta}_j^{-K}$. This then allows us to re-write (2.8) as $\exp\left(\langle\boldsymbol{z}_t^{-K},\boldsymbol{\theta}_j^{-K}\rangle-\psi(\boldsymbol{\theta}_j^{-K})\right)$, where the log-partition function $\psi\left(\boldsymbol{\theta}_j^{-K}\right)=\log(1/T_{jK})$.

Now we will show that the expectation parameter given by $\nabla\psi\left(\boldsymbol{\theta}_j^{-K}\right)$ is exactly the transition probability distribution corresponding to state j. From this we will be able to show the Legendre dual of the expectation parameter to be the negative entropy of the transition

distribution of j. To see this, note that the log-partition function may be written as

$$\psi\left(\boldsymbol{\theta}_j^{-K}\right) = \log\frac{1}{T_{jK}} = \log\left(\frac{T_{jK} + \sum_{k=1}^{K-1} T_{jk}}{T_{jK}}\right) = \log\left(1 + \sum_{k=1}^{K-1} e^{\theta_{jk}^{-K}}\right).$$

We can therefore write the expectation parameter as

$$\boldsymbol{m}_j = \nabla\psi(\boldsymbol{\theta}_j^{-K}) = \nabla\log\left(1 + \sum_{k=1}^{K-1} e^{\theta_{jk}^{-K}}\right) = \left\{\frac{e^{\theta_{jk}^{-K}}}{1 + \sum_{k=1}^{K-1} e^{\theta_{jk}^{-K}}}\right\}_{k=1}^{K-1} = \left\{T_{jk}\right\}_{k=1}^{K-1}.$$

Then the Legendre dual ϕ can be written as

$$\phi(\boldsymbol{m}_j) = \langle \boldsymbol{m}_j, \boldsymbol{\theta}_j^{-K}\rangle - \psi(\boldsymbol{\theta}_j^{-K}) = \sum_{k=1}^{K-1} T_{jk}\log\frac{T_{jk}}{T_{jK}} + \log T_{jK} = \sum_{k=1}^{K} T_{jk}\log T_{jk}.$$

Note that this is exactly the negative entropy of the transition probability distribution $\boldsymbol{\pi}_j = T_{j\cdot} = \left\{T_{jk}\right\}_{k=1}^{K}$.

The Bregman divergence derived from ϕ is therefore

$$\begin{aligned} d_\phi(\boldsymbol{z}_t, \boldsymbol{m}_j) &= \phi(\boldsymbol{z}_t) - \phi(\boldsymbol{m}_j) - \langle \boldsymbol{z}_t - \boldsymbol{m}_j, \nabla\phi(\boldsymbol{m}_j)\rangle \\ &= \sum_{k=1}^{K} z_{tk}\log z_{tk} - \sum_{k=1}^{K} T_{jk}\log T_{jk} - \sum_{k=1}^{K}(z_{tk} - T_{jk})\left(1 + \log T_{jk}\right) \\ &= \sum_{k=1}^{K} z_{tk}\log\frac{z_{tk}}{T_{jk}} = \mathrm{KL}(\boldsymbol{z}_t, \boldsymbol{m}_j), \end{aligned}$$

since we have $\sum_{k=1}^{K} z_{tk} = 1$ and $\sum_{k=1}^{K} T_{jk} = 1$.

Then, by the bijection established in Banerjee et al. [BMDG05], we can write the distribution

$\Pr(\boldsymbol{z}_t | z_{t-1,j} = 1)$ as

$$\Pr(\boldsymbol{z}_t | z_{t-1,j} = 1) = \exp\left(\left\langle \boldsymbol{z}_t^{-K}, \boldsymbol{\theta}_j^{-K}\right\rangle - \psi\left(\boldsymbol{\theta}_j^{-K}\right)\right) = \exp(-d_\phi(\boldsymbol{z}_t, \boldsymbol{m}_j))b_\phi(\boldsymbol{z}_t),$$

where the $\boldsymbol{m}$-independent function $b_\phi(\boldsymbol{z}_t)$ is

$$b_\phi(\boldsymbol{z}_t) = \exp(\phi(\boldsymbol{z}_t)) = \exp\left(\sum_{k=1}^{K} z_{tk}\log z_{tk}\right) = 1,$$

where we have again used the binary 1-of-K representation of $\boldsymbol{z}_t$. This completes the derivation of equation (2.2) in section 2.2.1.1 in Chapter 2.

We can now apply the recipe in [JKJ12], Lemma 3.1 to this Bregman divergence representation. In particular, the variance on an exponential family distribution may be scaled by appropriately scaling the partition function and the underlying natural parameter. Stated simply, we can introduce a new parameter $\hat{\beta}$, as in Jiang et al., and use a scaled version of the Bregman divergence representation, i.e., $\exp(-\hat{\beta} d_\phi(\boldsymbol{z}_t, \boldsymbol{m}_j)) b_{\hat{\beta}\phi}(\boldsymbol{z}_t)$.

2.6.2.1 EM Algorithm Asymptotics

Next we discuss the application of small-variance asymptotics to the Baum-Welch / EM algorithm for parametric hidden Markov models, and show its reduction to segmental K-means.

The reduction of the E-step posterior marginal update to the MAP estimate is fairly straightforward but involved. Let us define $\gamma(z_{tk})$ as the probability of the value of the t^{th} hidden variable being k, given the observations, the means, and the T matrix. This probability can be expressed as

$$\gamma(z_{tk}) = \frac{\alpha(z_{tk})\beta(z_{tk})}{\sum_{l=1}^{K} \alpha(z_{tl})\beta(z_{tl})} = \frac{1}{1 + \sum_{l=1; l\neq k}^{K} \frac{\alpha(z_{tl})\beta(z_{tl})}{\alpha(z_{tk}), \beta(z_{tk})}} \tag{2.9}$$

where $\alpha(\boldsymbol{z}_t) = p(\boldsymbol{x}_1, ..., \boldsymbol{x}_t, \boldsymbol{z}_t)$ and $\beta(\boldsymbol{z}_t) = p(\boldsymbol{x}_{t+1}, ..., \boldsymbol{x}_N \,|\, \boldsymbol{z}_t)$; the γ probabilities are computed in the E-step of the EM algorithm for the standard HMM. Asymptotically as $\sigma^2 \to 0$, we find that this probability goes to zero for all states k except the one that falls on the MAP path, which can be shown by expanding the ratios of α and β probabilities in the above denominator. Thus in the E-step analogue of our algorithm, one would first find the MAP path, and set the corresponding (temporal) posterior probability to 1 for each latent state on that path, and that for the other states to 0. Note that the MAP path for the Z variables

is obtained using a standard Viterbi dynamic programming algorithm with a forward and backward pass.

To complete the derivation of segmental k-means, we now consider the updates to T and $\boldsymbol{\mu}$ in the M-step analog. Recall the expected log-likelihood equation used in the M-step (see, e.g., Bishop [Bis06]):

$$\mathscr{Q}(\Theta,\Theta_{old}) = \sum_{k=1}^{K}\gamma(z_{1k})\ln\pi_k + \sum_{n=1}^{N}\sum_{k=1}^{K}\gamma(z_{nk})\ln p(\mathbf{x}_n|\boldsymbol{\mu}_k) + \sum_{n=2}^{N}\sum_{j=1}^{K}\sum_{k=1}^{K}\xi\left(z_{n-1,j},z_{nk}\right)\log T_{jk}.$$

Here $\xi\left(z_{n-1,j},z_{nk}\right)$ is the joint posterior marginal of pairwise latent variables, and Θ and Θ_{old} represent the new and existing values, respectively, of Z, $\boldsymbol{\mu}$ and T. We first consider maximizing $\mathscr{Q}(\Theta,\Theta_{old})$ with respect to the transition probabilities.

It can be noted that since we have hard cluster assignments in the asymptotic E-step, the joint posterior marginals will also be binary, and therefore $\sum_{n=1}^{N}\xi\left(z_{n-1,j},z_{nk}\right)$ represents the total number of times the transitions from state j to state k has occurred in the entire chain. Thus, for each state j, the maximization of the expected log-likelihood is a linear program over the $\log T_{jk}$s, with the solution being the empirical transition probabilities computed from the MAP transition sequence.

The final step is the updates to the means. This is similar to the derivation of the k-means cluster mean update as a limit of the M-step in a Gaussian mixture model, and can be straightforwardly derived to be an update of the empirical means of all points assigned to each state. This completes the derivation of segmental k-means as a small-variance approximation of the EM algorithm in the presence of exponential family probabilities.

2.6.3 Algorithm for the infinite Hidden Markov Model

Here, we give the detailed description of the algorithm in Section 3.2. The optimization problem we aim to solve is

$$\min_{K,z,\boldsymbol{\mu},T} \quad \sum_{t=1}^{N}\|\boldsymbol{x}_t-\boldsymbol{\mu}_{z_t}\|^2 - \lambda\sum_{t=2}^{N}\log T_{z_{t-1},z_t} + \lambda_1\sum_{k=1}^{K}(s_k-1) + \lambda_2(K-1), \tag{2.10}$$

where $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ are the observations, $\{z_1, \ldots, z_N\}$ is the hidden state sequence, K is the total number of hidden states, T is the transition probability matrix with $T_{i,j} = \Pr(z_{t+1} = j | z_t = i)$, and $s_i (i = 1, \ldots, K)$ is the number of states that can be reachable from state i. Note that we are expressing the transition penalties here in terms of T as opposed to the KL divergence, for ease of presentation.

To optimize (2.10), we follow an alternating minimization framework. We first determine the sequence of states to optimize the objective when all but Z is fixed using a forward-backward routine. Here, we cannot apply exact dynamic programming, due to the possible creation of new states as well as the change in transition probabilities when either creating a new state or paying a λ_1 penalty. We then update the means of each state as the empirical means based on the state assignments, and the transition matrix as the empirical transition matrix. We further adopt a move analogous to that described for the hard HDP in [KJ12]. This step determines if the objective will decrease if we create a new hidden state in a certain fashion; in particular, for each existing state j, we compute the change in objective that occurs when observations that transition from state j to some state k are given their own new hidden state.

Specifically, given observations $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}, \lambda, \lambda_1, \lambda_2$, our high-level algorithm proceeds as follows:

1. Initialization: initialize with one hidden state. The parameters are therefore $K = 1, \boldsymbol{\mu}_1 = \frac{1}{N} \sum_{i=1}^{N} \boldsymbol{x}_i, T = 1$.

2. Perform a forward-backward step (via approximate dynamic programming) to update Z.

3. Update $K, \boldsymbol{\mu}, T$.

4. For each state i, ($i = 1, \ldots, K$), check if the set of observations to any state j that are reached by transitioning out of i can form a new dedicated hidden state and lower the objective function in the process. If there are several such moves, choose the one with the maximum improvement in objective function.

5. Update $K, \boldsymbol{\mu}, T$.

6. Iterate steps (2)-(5) until convergence.

Note: Step (3) here is only for clearer presentation of step (4).

2.6.3.1 Forward-Backward Step

In the forward-backward step, we will compute a $K \times N$ matrix α, where $\alpha(c,t)$ represents the minimum cost over paths of length t from the beginning of the sequence and that reach state c at time step t. We use the term "cost" to refer to the sum of the distances of points to state means, as well as the additive penalties incurred. Since we are interested in the possibility of potentially creating new states during this forward-backward process, and the creation of new states will necessarily change the transition probabilities, it does not appear that α can be computed exactly. We instead describe a procedure that computes an upper bound for each value of α.

To give further intuition for why it is difficult to compute the exact value of α: suppose we have computed the minimum cost of paths up to step $t-1$ and we would like to compute the values of α for step t. The value of a path that ends in state c is obtained by examining, for all states i, the cost of a path that ends at i at step $t-1$ and then transitions to state c at step t. Thus, we must consider the transition from i to c. If there are existing transitions from state i to state c, then we proceed as in a standard forward-backward algorithm. However, we are also interested in two other cases—one where there are no existing transitions from i to c but we consider this transition along with a penalty λ_1, and another where an entirely new

state is formed and we pay a penalty λ_2. In the first case, the standard forward-backward routine faces an immediate problem, since when we try to compute the cost of the path given by $\alpha(c,t)$, the cost will be infinite as there is a $-\log(0)$ term from the transition probability. We must therefore alter the forward-backward routine, or there will never be new states created nor transitions to an existing state which previously had no transitions. The main idea is to derive and use bounds on how much the transition matrix can change under the above scenarios. As long as we can show that the values we obtain for α are upper bounds, then we can show that the objective function will decrease after the forward-backward routine, as the existing sequence of states is also a valid path (with no new incurred penalties). That is, the cost we compute here is an upper bound of both the non-penalized objective function value and newly-introduced penalties. If there is no new states created or no new transitions happened, the upper bound for the newly-introduced penalties is zero as mentioned before.

Now we describe the algorithm in more detail. In the following description, old_K is the number of states from the previous iteration, and K is the number of states before the computation of α at time step t. Then, for each state $1 \leq c \leq K$,

1. If we are transitioning from a state $i \leq old\_K$,

   - Standard situation: If $c \leq old\_K$ and $T_{i,c} \neq 0$, we compute (as in the parametric case):

   $$d(i,c) = \|\boldsymbol{x}_t - \boldsymbol{\mu}_c\|^2 - \lambda \times \log(T_{i,c}). \tag{2.11}$$

   - No existing transitions: Otherwise, we use the following upper bound when transitioning to state c:

   $$d(i,c) = \|\boldsymbol{x}_t - \boldsymbol{\mu}_c\|^2 - \lambda \times \log\left(\frac{1}{n_i}\right) + \lambda \times E_i + \lambda_1, \tag{2.12}$$

   where,

- n_i is the total number of transitions out of state i.
- E_i is the upper bound of the possible change in one transition probability from state i incurred by adding one state to the reachable pool of state i. We compute the change of the maximum of $\frac{n_{ij}}{n_i}$, and we have (assume j is the largest one)

$$E_i = (n_{ij} - 1) \times \left(\log \frac{n_{ij}}{n_i} - \log \frac{n_{ij} - 1}{n_i} \right). \tag{2.13}$$

- λ_1 is the penalty incurred from transiting to a new state.

2. Entirely new state: Otherwise, we use the bound:

$$d(i,c) = \|\boldsymbol{x}_t - \boldsymbol{\mu}_c\|^2 - \lambda \times \log \left(\frac{1}{N-1} \right) + \lambda_1. \tag{2.14}$$

where,

- $\frac{1}{N-1}$ is an upper bound of the transition probability from state i to state c,
- 0 is the upper bound of possible change in the transition probabilities from state k where the original transition occurred for this transition. Here, in this path which involves transition from an entirely new state i to state c, we not only add a new row to the transition matrix T, but also change another existing row of T. That is, the transit-out probability of state k will be changed since it loses one count of transition to state c.
- λ_1 is the penalty incurred when adding this new state transition out of state i.

Empirically, we will use $d(i,c) = \|x_t - \mu_c\|^2 + \lambda_1$ since it still decreases the objective function value monotonically most of the time in practice and yields better results.

3. We compute the minimum among all states:

$$\alpha(c,t) = \min_{1 \le i \le K} \alpha(i, t-1) + d(i,c) \tag{2.15}$$

To check if this time step can be created as a new hidden state, we find

$$d_{\min} = \min_{i,c} d(i,c). \tag{2.16}$$

If $d_{\min} > \lambda_1 + \lambda_2$, we create a new hidden state (this is the penalty incurred for transiting to a new state and a new hidden state). We let $K = K + 1$,

$$\alpha(K,t) = \min_{1 \le k \le K-1} \alpha(k,t-1) + \lambda_1 + \lambda_2. \tag{2.17}$$

This follows from the description of (II).

Now, we prove the correctness of the upper bounds. For the first bound, when we consider transitioning from i to c such that $T_{ic} = 0$, we are adding a new transition out of state i, which then impacts the other transition probabilities out of state i. Thus, the E_i bound determines how much the change in the other transition probabilities impacts the current path cost.

Lemma 1. *E_i is an upper bound of the possible change in the objective function value, in terms of other transition probabilities from state i, incurred by adding one transition to a new state in Step (I). Here, we assume the total number of transitions from state i is fixed.*

Proof. Denote n_i as the total number of transitions from state i, n_{ij} as the total number of transitions from state i to state j with $n_{ij} > 0$. Thus, the possible change for other transitions is

$$(n_{ij} - 1) \times \left(\log \frac{n_{ij}}{n_i} - \log \frac{n_{ij} - 1}{n_i} \right) \ge 0.$$

Let $f(x) = (x-1)(\log x - \log(x-1)), x \ge 2$, we have

$$\begin{aligned} f'(x) &= \log x - \log(x-1) - \frac{1}{x} \\ &= \log x - [\log x + \frac{-1}{x} - \frac{1}{2x^2} + o(\frac{1}{2x^2})] - \frac{1}{x} \\ &= \frac{1}{2x^2} + o(\frac{1}{2x^2}) > 0 \end{aligned}$$

Thus, $f(x)$ is increasing as x increases for $x \geq 2$. When $x = 1$, we have by definition $f(1) = 0$. Therefore,

$$E_i = (n_{ik} - 1) \times \left(\log \frac{n_{ik}}{n_i} - \log \frac{n_{ik} - 1}{n_i} \right) = \max_j (n_{ij} - 1) \times \left(\log \frac{n_{ij}}{n_i} - \log \frac{n_{ij} - 1}{n_i} \right),$$

where $k = \text{argmax}_j n_{ij}$. □

The second bound deals with transitioning from an entirely new state. We are adding one row to the transition matrix T, which then also changes another row of transition probabilities out of state k where this transition previously is coming from state k.

Lemma 2. *0 is an upper bound of possible change in the objective function value in terms of the transition probabilities from state k where the original transition occurred for this transition in Step (II).*

Proof. Denote n_k the total number of transitions from state k, n_{kj} the total number of transitions from state k to state j with $n_{kj} > 0$. Without loss of generality, we assume that the lost transitions are all from n_{ki} and the number of lost transitions is x. Thus, the change is

$$\begin{aligned} &\sum_j n_{kj} \log \frac{n_{kj}}{n_k} - \sum_{j \neq i} n_{kj} \log \frac{n_{kj}}{n_k - x} - (n_{ki} - x) \log \frac{n_{ki} - x}{n_k - x} \\ &= \sum_{j \neq i} n_{kj} \log \frac{n_k - x}{n_k} + n_{ki} \left(\log \frac{n_{ki}}{n_k} - \log \frac{n_{ki} - x}{n_k - x} \right) + x \log \frac{n_{ki} - x}{n_k - x} \leq 0, \end{aligned}$$

since $\frac{n_{ki}}{n_k} \leq 1, \frac{n_k - x}{n_k} < 1, \frac{n_{ki} - x}{n_k - x} \leq 1$, and $\frac{n_{ki}}{n_k} \geq \frac{n_{ki} - x}{n_k - x}$. □

From Lemma 1 and 2, we know that in each time step we compute an upper bound of the minimum non-penalized objective function value. And since we add λ_1 and λ_2 in α whenever we transition to a state where there was no existing transitions or we create an entirely new state respectively, we also manage to upper-bound the newly-introduced penalties. Combining these two together, we have the following:

Proposition 1. *The computation of α gives an upper bound of the minimum cost of every possible path.*

2.6.3.2 Local Move - Step 4

After finishing the forward-backward step and updating the means and transition matrix, we check locally if we should create a new state by determining, for all i and k, if the objective function is lowered when all data points in state k that reached state k via state i are put into a new state. In particular, we determine and execute the single best such move.

In detail, for each state $1 \leq i \leq K$, we consider all the time steps A_i that are one step from a time step with state i. These time steps A_i can be grouped by their states: $A_i(1), \ldots, A_i(K)$, where $A_i(j)$ indicates time steps belonging to state j. Then, for all states j with $|A_i(j)| > 0$, we can compute the objective function contribution from these time steps. We have

$$old = \sum_{t \in A_i(j)} \{\|\boldsymbol{x}_t - \boldsymbol{\mu}_{z_t}\|^2 - \lambda \times \log T_{i,z_t} - \lambda \times \log T_{z_t,z_{t+1}}\}. \tag{2.18}$$

If we let $A_i(j)$ be a new hidden state, the contribution would be

$$new = \sum_{t \in A_i(j)} \{\|\boldsymbol{x}_t - \bar{\boldsymbol{x}}_t\|^2 - \lambda \times \log T_{i,z_t} - \lambda \times \log T_{K+1,z_{t+1}}\} + \lambda_1 \times K_i(j) + \lambda_2, \tag{2.19}$$

where

- $\bar{\boldsymbol{x}}_t = \frac{1}{|A_i(j)|} \sum_{t \in A_i(j)} \boldsymbol{x}_t$;
- Let $B_i(j)$ be the time steps which are one-step from those of $A_i(j)$. $T_{K+1,z_{t+1}}$ is just the empirical transition probability from time step $t \in A_i(j)$ to the next time step $t+1 \in B_i(j)$.
- $K_i(j)$ is the number of states in $B_i(j)$.
- Since there is no change of λ_1 penalty for state i, we do not need to consider this in the "old" contribution.

If $new < old$, then $A_i(j)$ is considered as a candidate for a new hidden state.

After a whole pass of the states, we find the largest reduction

$$[i, j] \in \text{argmax}_{i,j}\ old_i(j) - new_i(j), \tag{2.20}$$

and create $A_i(j)$ as a new hidden state.

In summary, in the forward-backward step, the cost we compute is an upper bound of both the non-penalized objective function value and newly-introduced penalties. If there is no new states created or no new transitions happened, the upper bound for the newly-introduced penalties is zero. That is, we preserve the non-penalized objective function value of any path from previous iteration, and upper-bound the non-penalized objective function value and the newly-introduced penalties of any new path we find. In the local move step, we further reduce the objective function value by considering locally for each state we get in the forward-backward step. Therefore, we have

Proposition 2. *The algorithm decreases the objective function value in each iteration.*

Proof. We know that the best path old_P obtained from last iteration would be a possible path in this iteration, and from the forward-backward step, it would preserve its cost from the previous iteration. Thus, we have

$$\alpha(new\_P) = \min_{\text{all possible P}} \alpha(P) \leq \alpha(old\_P).$$

From Proposition 1, we know α gives the upper bound of the additional incurred cost, thus

$$\text{cost}(new\_P) \leq \text{cost}(old\_P).$$

We also further decreases the cost from the local move, which preserves the inequality. That is, we have

$$\text{cost}(new\_P) \leq \text{cost}(old\_P).$$

□

2.6.4 Additional Experimental Details

2.6.4.1 Heuristic for Parameter Selection

In this section, we discuss a simple but empirically promising heuristic for parameter selection. While performing a grid-search over candidate λ values, the number of learned states does not collapse to the degenerate case with a uniform step size of 1; there is always a gap between the trivial case of 1 hidden state, and the next higher count. We consider this smallest number of states larger than 1 as the estimate of state count learned by our algorithm, and select parameters that yield this number. For multiple parameter combinations that yield the estimated number of states, we choose the one which gives the best fit, measured in terms of the smallest non-penalized objective function value. We stress that this technique is a heuristic, and do not claim any theoretical justification for it.

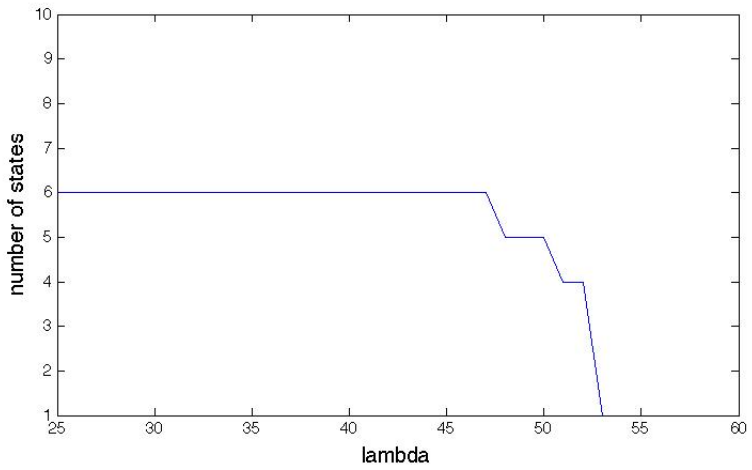


Figure 2.4: An example showing the trend of the number of states as a function of lambda.

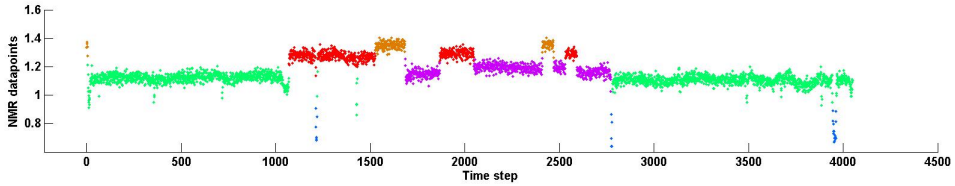


Figure 2.5: Visual display of the state assignments provided by the asymp-HMM algorithm on the well-log data. Each color shows a different state (best viewed in color); see text for details.

2.6.4.2 Well-log data

We provide some additional experimental results that illustrate the qualitative performance of our algorithms on a changepoint detection problem. The data consists of 4050 noisy NMR measurements of rock strata obtained via lowering a probe through a bore-hole. The data has been previously analyzed in [RF96] by eliminating the forty greatest outliers and running a changepoint detection algorithm with a fixed number of changepoints. This approach works well as this one-dimensional dataset can be inspected visually to make a decision on whether to throw away datapoints and get a rough idea for the number of changepoints. It has been noted in [RF96] that Gaussian mixtures seem to be a good model for this data; in addition visual inspection of the data seems to suggest five clusters, at five distinct depths. Thus we used a five-state Gaussian HMM to model this data. As in our synthetic experiments, we first ran k-means for initialization of the mixture means and randomly initialized the transition/pseudo-transition kernels.

The state sequence inferred by our algorithm is color-plotted in Figure 2.5. The algorithm seems to have detected the changepoints accurately, with imperceptibly small noise. Relative performance trends are similar to the synthetic case; the standard HMM took an average of 16 iterations to converge, with an average running time of 4.5s. Our algorithm converges in 9 iterations, with an average time of 1.7s with $\lambda = 3$.

Chapter 3: Gamma Processes, Stick-Breaking, and Variational Inference

The gamma process is a versatile pure-jump Lévy process with widespread applications in various fields of science. Of late it is emerging as an increasingly popular prior in the Bayesian nonparametric literature within the machine learning community; it has recently been applied to exchangeable models of sparse graphs [CF13] as well as for nonparametric ranking models [CTM13]. It also has been used as a prior for infinite-dimensional latent indicator matrices [Tit08]. This latter application is one of the earliest Bayesian nonparametric approaches to count modeling, and as such can be thought of as an extension of the venerable Indian Buffet Process to modeling latent structures where each feature can occur multiple times for a datapoint, instead of being simply binary.

The flexibility of gamma process models allows them to be applied in a wide variety of Bayesian nonparametric settings, but their relative complexity makes principled inference nontrivial. In particular, most direct applications of the gamma process in the Bayesian nonparametric literature use Markov chain Monte Carlo samplers (typically Gibbs sampling) for posterior inference, which often suffers from poor scalability. For other Bayesian nonparametric models—in particular those involving the Dirichlet process or beta process—a successful thread of research has considered variational alternatives to standard sampling methods [BJ03, TKW07, WPB11]. One first derives an explicit construction of the underlying "weights" of the atomic measure component of the random measures underlying the

infinite priors; so-called "stick-breaking" processes for the Dirichlet and beta processes yield such a construction. Then these weights are truncated and integrated into a mean-field variational inference algorithm. For instance, stick-breaking was derived for the Dirichlet process in the seminal paper by Sethuraman [Set94], which was in turn used for variational inference in Dirichlet process models [BJ03]. Similar stick-breaking representations for a special case of the Indian Buffet Process [TGG07] and the beta process [PZW$^+$10] have been constructed, and have naturally led to mean-field variational inference algorithms for nonparametric models involving these priors [DVMGT09, PCB11]. Such variational inference algorithms have been shown to be more scalable than the sampling-based inference techniques normally used; moreover they work with the full model posterior without marginalizing out any variables.

In this chapter we propose a variational inference framework for gamma process priors using a novel stick-breaking construction of the process. We use the characterization of the gamma process as a *completely random measure* (CRM), which allows us to leverage Poisson process properties to arrive at a simple derivation of the rate measure of our stick-breaking construction, and show that it is indeed equal to the Lévy measure of the gamma process. We also use the Poisson process formulation to derive a bound on the error of the truncated version compared to the full process, analogous to the bounds derived for the Dirichlet process [IJ01], the Indian Buffet Process [DVMGT09] and the beta process [PCB11]. We then, as a particular example, focus on the infinite Gamma-Poisson model of [Tit08] (note that variational inference need not be limited to this model). This model is a prior on infinitely wide latent indicator matrices with non-negative integer-valued entries; each column has an associated parameter independently drawn from a gamma distribution, and the matrix values are independently drawn from Poisson distributions with these parameters as means. We develop a mean-field variational technique using a truncated version of our

stick-breaking construction, and a sampling algorithm that uses Monte Carlo integration for parameter marginalization, similar to [PZW+10], as a baseline inference algorithm for comparison. We also derive a variational algorithm based on the naïve construction of the gamma process. Finally we compare these with variational algorithms based on Beta-Bernoulli priors on a non-negative matrix factorization task involving the Psychological Review, NIPS, KOS and New York Times document corpora, and show that the variational algorithm based on our construction performs and scales better than all the others.

Related Work. To our knowledge this is the first explicit "stick-breaking"-like construction of the gamma CRM, apart from the naïve approach of denormalizing the construction of the DP with a suitable gamma random variable [Mil11], [GRRB14]; moreover, as mentioned above, we develop a variational inference algorithm using the naïve construction (see 3.4.1) and show that it performs worse than our main algorithm on both synthetic and real datasets. The very general inverse Lévy measure algorithm of [WI98] requires inversion of the exponential integral, as does the generalized CRM construction technique of [OW12] when applied to the gamma process; since the closed form solution of the inverse of an exponential integral is not known, these techniques do not give us an analytic construction of the weights, and hence cannot be adapted to variational techniques in a straightforward manner. Other constructive definitions of the gamma process include [Thi08], who discusses a sampling-based scheme for the weights of a gamma process by sampling from a Poisson process. As an alternative to gamma process-based models for count modeling, recent research has examined the negative binomial-beta process and its variants [ZC12, ZHDC12, BMPJ14]; the stick-breaking construction of [PZW+10] readily extends to such models since they have beta process priors. The beta stick-breaking construction has also been used for variational inference in beta-Bernoulli process priors [PCB11], though they have scalability issues

when applied to the count modeling problems addressed in this work, as we show in the experimental section.

3.1 Background

3.1.1 Completely random measures

A completely random measure [Kin67, Jor10] $\mathbb{G}$ on a space $(\Omega, \mathscr{F})$ is defined as a stochastic process on $\mathscr{F}$ such that for any two disjoint Borel subsets $\mathscr{A}_1$ and $\mathscr{A}_2$ in $\mathscr{F}$, the random variables $\mathbb{G}(\mathscr{A}_1)$ and $\mathbb{G}(\mathscr{A}_2)$ are independent. The canonical way of constructing a completely random measure $\mathbb{G}$ is to first take a σ-finite product measure H on $\Omega \otimes \mathbb{R}^+$, then draw a countable set of points $\{(\omega_k, p_k)\}$ from a Poisson process on a Borel σ-algebra on $\Omega \otimes \mathbb{R}^+$ with H as the rate measure. Then the CRM is constructed as $\mathbb{G} = \sum_{k=0}^{\infty} p_k \delta_{\omega_k}$, where the measure given to a measurable Borel set $B \subset \Omega$ is $\mathbb{G}(B) = \sum_{k:\omega_k \in B} p_k$. In this notation p_k are referred to as weights and the ω_k as atoms.

If the rate measure is defined on $\Omega \otimes [0,1]$ as $H(d\omega, dp) = cp^{-1}(1-p)^{c-1}B_0(d\omega)dp$, where B_0 is an arbitrary finite continuous measure on Ω and c is some constant (or function of ω), then the corresponding CRM constructed as above is known as a beta process. If the rate measure is defined as $H(d\omega, dp) = cp^{-1}e^{-cp}G_0(d\omega)dp$, with the same restrictions on c and G_0, then the corresponding CRM constructed as above is known as the gamma process. The total mass of the gamma process $G, G(\Omega)$, is distributed as $\text{Gamma}(cG_0(\Omega), c)$. The improper distributions in these rate measures integrate to infinity over their respective domains, ensuring a countably infinite set of points in a draw from the Poisson process. For the beta process, the weights p_k are in $[0,1]$, whereas for the gamma process they are in $[0,\infty)$. In both cases however the sum of the weights is finite, as can be seen from Campbell's theorem [Kin67], and is governed by c and the total mass of the base measure on Ω. For completeness we note that completely random measures as defined in [Kin67]

have three components: a set of fixed atoms, a deterministic measure (usually assumed absent), and a random discrete measure. It is this third component that is explicitly generated using a Poisson process, though the fixed component can be readily incorporated into this construction [Kin93].

If we create an atomic measure by normalizing the weights $\{p_k\}$ from the gamma process, i.e. $D = \sum_{k=0}^{\infty} \pi_k \delta_{\omega_k}$ where $\pi_k = p_k / \sum_{i=0}^{\infty} p_i$, then D is known as a *Dirichlet process* [Fer73], denoted as $D \sim \mathrm{DP}(\alpha_0, H_0)$ where $\alpha_0 = G_0(\Omega)$ and $H_0 = G_0/\alpha_0$. It is not a CRM as the random variables induced on disjoint sets lack independence because of the normalization; it belongs to the class of normalized random measures with independent increments (NRMIs).

3.1.2 Stick-breaking for the Dirichlet and Beta Processes

A recursive way to generate the weights of random measures is given by stick-breaking, where a unit interval is subdivided into fragments based on draws from suitably chosen distributions. For example, the stick-breaking construction of the Dirichlet process [Set94] is given by

$$D = \sum_{i=1}^{\infty} V_i \prod_{j=1}^{i-1} (1 - V_j) \delta_{\omega_i},$$

where $V_i \overset{iid}{\sim} \mathrm{Beta}(1, \alpha), \quad \omega_i \overset{iid}{\sim} H_0$. Here the length of the first break from a unit-length stick is given by V_1. In the next round, a fraction V_2 of the remaining stick of length $1 - V_1$ is broken off, and we are left with a piece of length $(1 - V_2)(1 - V_1)$. The length of the piece in the next round is therefore given by $V_3(1 - V_2)(1 - V_1)$, and so on. Note that the weights belong to $(0, 1)$, and since this is a normalized measure, the weights sum to 1 almost surely. This is consistent with the use of the Dirichlet process as a prior on probability distributions.

This construction was generalized in [PZW+10] to yield stick-breaking for the beta process:

$$B = \sum_{i=1}^{\infty} \sum_{j=1}^{C_i} V_{ij}^{(i)} \prod_{l=1}^{i-1} (1 - V_{ij}^{(l)}) \delta_{\omega_{ij}}, \tag{3.1}$$

where $V_{ij}^{(i)} \overset{iid}{\sim} \text{Beta}(1,\alpha)$, $C_i \overset{iid}{\sim} \text{Poisson}(\gamma)$, $\omega_{ij} \overset{iid}{\sim} \frac{1}{\gamma} B_0$. We use this representation as the basis for our stick breaking-like construction of the Gamma CRM, and use Poisson process-based proof techniques similar to [PBJ12] to derive the rate measure.

3.2 The Stick-breaking Construction of the Gamma Process

3.2.1 Constructions and proof of correctness

We propose a simple recursive construction of the gamma process CRM, based on the stick-breaking construction for the beta process proposed in [PZW+10, PBJ12]. In particular, we augment (or 'mark') a slightly modified stick-breaking beta process with an independent gamma-distributed random measure and show that the resultant Poisson process has the rate measure $H(d\omega, dp) = cp^{-1}e^{-cp}G_0(d\omega)dp$ as defined above. We show this by directly deriving the rate measure of the marked Poisson process using product distribution formulae. Our proposed stick-breaking construction is as follows:

$$G = \sum_{i=1}^{\infty} \sum_{j=1}^{C_i} G_{ij}^{(i)} V_{ij}^{(i)} \prod_{l=1}^{i} (1 - V_{ij}^{(l)}) \delta_{\omega_{ij}}, \tag{3.2}$$

where $G_{ij}^{(i)} \overset{iid}{\sim} \text{Gamma}(\alpha+1, c)$, $V_{ij}^{(i)} \overset{iid}{\sim} \text{Beta}(1,\alpha)$, $C_i \overset{iid}{\sim} \text{Poisson}(\gamma)$, $\omega_{ij} \overset{iid}{\sim} \frac{1}{\gamma} H_0$. As with the beta process stick-breaking construction, the product of beta random variables allows us to interpret each j as corresponding to a stick that is being broken into an infinite number of pieces. Note that the expected weight on an atom in round i is $\alpha^i / c(1+\alpha)^i$. The parameter c can therefore be used to control the weight decay cadence along with α.

The above representation provides the clearest view of the construction, but is somewhat cumbersome to deal with in practice, mostly due to the introduction of the additional gamma random variable. We reduce the number of random variables by noting that the product of a $\text{Beta}(1,\alpha)$ and a $\text{Gamma}(\alpha+1, c)$ random variable has an $\text{Exp}(c)$ distribution; we also

perform a change of variables on the product of the $(1-V_{ij})$s to arrive at the following equivalent construction, for which we now prove its correctness:

Theorem 1. *A gamma CRM with positive concentration parameters α and c and finite base measure H_0 may be constructed as*

$$G = \sum_{i=1}^{\infty} \sum_{j=1}^{C_i} E_{ij} e^{-T_{ij}} \delta_{\omega_{ij}}, \tag{3.3}$$

where $E_{ij} \overset{iid}{\sim} Exp(c), \quad T_{ij} \overset{ind}{\sim} Gamma(i,\alpha), \quad C_i \overset{iid}{\sim} Poisson(\gamma), \quad \omega_{ij} \overset{iid}{\sim} \frac{1}{\gamma} H_0.$

Proof. Note that, by construction, in each round i in (3.3), each set of weighted atoms $\{(\omega_{ij}, E_{ij}e^{-T_{ij}})\}_{j=1}^{C_i}$ forms a Poisson point process since the C_i are drawn from a Poisson(γ) distribution. In particular, each of these sets is a *marked* Poisson process [Kin93], where the atoms ω_{ij} of the Poisson process on Ω are marked with the random variables $E_{ij}e^{-T_{ij}}$ that have a probability measure on $(0,\infty)$. The superposition theorem of [Kin93] tells us that the countable union of Poisson process is itself a Poisson process on the same measure space; therefore denoting $G_i = \sum_{j=1}^{C_i} E_{ij} e^{-T_{ij}} \delta_{\omega_{ij}}$, we can say $G = \bigcup_{i=1}^{\infty} G_i$ is a Poisson process on $\Omega \times [0,\infty)$. We show below that the rate measure of this process equals that of the Gamma CRM.

Now, we note that the random variable $E_{ij}e^{-T_{ij}}$ has a probability measure on $[0,\infty)$; denote this by q_{ij}. We are going to mark the underlying Poisson process with this measure. The density corresponding to this measure can be readily derived using product distribution formulae. To that end, ignoring indices, if we denote $W = \exp(-T)$, then we can derive its distribution by a change of variable. Then, denoting $Q = E \times W$ where $E \sim \text{Exp}(c)$, we can use the product distribution formula to write the density of Q as

$$f_Q(q) = \int_0^1 \frac{\alpha^i}{\Gamma(i)} (-\log w)^{i-1} w^{\alpha-2} c e^{-c\frac{q}{w}} \mathrm{d}w,$$

where $T \sim \text{Gamma}(i, \alpha)$. Formally speaking, this is the Radon-Nikodym density corresponding to the measure q, since it is absolutely continuous with respect to the Lebesgue measure on $[0,\infty)$ and σ-finite by virtue of being a probability measure. Furthermore, these conditions hold for all the measures that we have in our union of marked Poisson processes; this allows us to write the density of the combined measure as

$$\begin{aligned}
f(p) &= \sum_{i=1}^{\infty} \int_0^1 \frac{\alpha^i}{\Gamma(i)} (-\log w)^{i-1} w^{\alpha-2} c e^{-c\frac{p}{w}} \mathrm{d}w \\
&= \int_0^1 \sum_{i=1}^{\infty} \frac{\alpha^i}{\Gamma(i)} (-\log w)^{i-1} w^{\alpha-2} c e^{-c\frac{p}{w}} \mathrm{d}w \\
&= \int_0^1 \alpha w^{-2} c e^{-c\frac{p}{w}} \mathrm{d}w \\
&= \alpha p^{-1} e^{-cp} \\
&= c p^{-1} e^{-cp} \frac{\alpha}{c},
\end{aligned}$$

where we have used monotone convergence to get the Taylor expansion of $\exp(-\alpha \log w)$ inside the integral. Note that the measure defined on $\mathscr{B}([0,\infty))$ by the "improper" gamma distribution $p^{-1}e^{-cp}$ is σ-finite, in the sense that we can decompose $[0,\infty)$ into the countable union of disjoint intervals $[1/k, 1/(k-1)), \quad k = 1, 2, \ldots \infty$, each of which has finite measure. In particular, the measure of the interval $[1,\infty)$ is given by the exponential integral.

Therefore the rate measure of the process G as constructed here is $G(d\omega, dp) = cp^{-1}e^{-cp} G_0(d\omega) dp$ where G_0 is the same as H_0 up to the multiplicative constant $\frac{\alpha}{c}$, and therefore satisfies the finiteness assumption imposed on H_0. □

We use the form specified in the theorem above in our variational inference algorithm since the variational distributions on almost all the parameters and variables in this construction lend themselves to simple closed-form exponential family updates. As an aside, we note that the random variables $(1 - V_{ij})$ have a $\text{Beta}(\alpha, 1)$ distribution; therefore if we

denote $U_{ij} = 1 - V_{ij}$ then the construction in (3.2) is equivalent to

$$G = \sum_{i=1}^{\infty} \sum_{j=1}^{C_i} E_{ij}^{(i)} \prod_{l=1}^{i} U_{ij}^{(l)} \delta_{\omega_{ij}},$$

where $E_{ij}^{(i)} \overset{iid}{\sim} \text{Exp}(c)$, $U_{ij}^{(i)} \overset{iid}{\sim} \text{Beta}(\alpha, 1)$, $C_i \overset{iid}{\sim} \text{Poisson}(\gamma)$, $\omega_{ij} \overset{iid}{\sim} \frac{1}{\gamma} H_0$. This notation therefore relates our construction to the stick-breaking construction of the Indian Buffet Process [TGG07], where the Bernoulli probabilities π_k are generated as products of iid $\text{Beta}(\alpha, 1)$ random variables : $\pi_1 = \nu_1$, $\pi_k = \prod_{i=1}^{k} \nu_i$ where $\nu_i \overset{iid}{\sim} \text{Beta}(\alpha, 1)$. In particular, we can view our construction as a generalization of the IBP stick-breaking, where the stick-breaking weights are multiplied with independent $\text{Exp}(c)$ random variables, with the summation over j providing an explicit Poissonization.

3.2.2 Truncation analysis

The variational algorithm requires a truncation level for the number of atoms for tractability. Therefore we need to analyze the closeness between the marginal distributions of the data drawn from the full prior and the truncated prior, with the stick-breaking prior weights integrated out. Our construction leads to a simpler truncation analysis if we truncate the number of rounds (indexed by i in the outer sum), which automatically truncates the atoms to a finite number. For this analysis, we will use the stick-breaking gamma process as the base measure of a Poisson likelihood process, which we denote by PP; this is precisely the model for which we develop variational inference in the next section. If we denote the gamma process as $G = \sum_{k=0}^{\infty} g_k \delta_{\omega_k}$, with g_k as the recursively constructed weights, then PP can be written as $PP = \sum_{k=0}^{\infty} p_k \delta_{\omega_k}$ where $p_k = \text{Poisson}(g_k)$. Under this model, we can obtain the following result, which is analogous to error bounds derived for other nonparametric models [IJ01, DVMGT09, PCB11] in the literature.

Theorem 2. *Let N samples $\boldsymbol{X} = (X_1, .., X_N)$ be drawn from $PP(G)$. If $G \sim \Gamma P(c, G_0)$, the full gamma process, then denote the marginal density of $\boldsymbol{X}$ as $\boldsymbol{m}_\infty(\boldsymbol{X})$. If G is a gamma process truncated after R rounds, denote the marginal density of $\boldsymbol{X}$ as $\boldsymbol{m}_R(\boldsymbol{X})$. Then*

$$\frac{1}{4}\int |\boldsymbol{m}_\infty(\boldsymbol{X}) - \boldsymbol{m}_R(\boldsymbol{X})| d\boldsymbol{X} \leq 1 - \exp\left\{-N\gamma\frac{\alpha}{c}\left(\frac{\alpha}{1+\alpha}\right)^R\right\}.$$

Proof. The starting intuition is that if we truncate the process after R rounds, then the error in the marginal distribution of the data will depend on the probability of positive indicator values appearing for atoms after the R^{th} round in the infinite version. Combining this with ideas analogous to those in [IJ00] and [IJ01], we get the following bound for the difference between the marginal distributions:

$$\frac{1}{4}\int |\mathbf{m}_\infty(\mathbf{X}) - \mathbf{m}_R(\mathbf{X})| d\mathbf{X} \leq \mathbb{P}\left\{\exists(k,j), k > \sum_{r=1}^{R} C_r, 1 \leq n \leq N \text{ s.t. } X_n(\omega_{kj}) > 0\right\}.$$

Since we have a Poisson likelihood on the underlying gamma process, this probability can be written as

$$\mathbb{P}(\cdot) = 1 - \mathbb{E}\left[\mathbb{E}\left\{\left(\prod_{r=R+1}^{\infty}\prod_{j=1}^{C_r} e^{-\pi_{rj}}\right)^N \middle| C_r\right\}\right],$$

where $\pi_{rj} = G_{rj}^{(r)} V_{rj}^{(r)} \prod_{l=1}^{r}(1 - V_{rj}^{(l)})$. We may then use Jensen's inequality to bound it as follows:

$$\begin{aligned}
\mathbb{P}(\cdot) &\leq 1 - \exp\left[N\sum_{r=R+1}^{\infty}\mathbb{E}\left\{\sum_{j=1}^{C_r}\log(e^{-\pi_{rj}})\right\}\right] \\
&= 1 - \exp\left[N\gamma\frac{1}{c}\sum_{r=R+1}^{\infty}\left(\frac{\alpha}{1+\alpha}\right)^r\right] \\
&= 1 - \exp\left\{-N\gamma\frac{\alpha}{c}\left(\frac{\alpha}{1+\alpha}\right)^R\right\}.
\end{aligned}$$

□

3.3 Variational Inference

As discussed in Section 3.2.2, we will focus on the infinite Gamma-Poisson model, where a gamma process prior is used in conjunction with a Poisson likelihood function. When integrating out the weights of the gamma process, this process is known to yield a nonparametric prior for sparse, infinite count matrices [Tit08]. We note that our approach should easily be applicable to other models involving gamma process priors.

3.3.1 The Model

To effectively perform variational inference, we re-write G as a single sum of weighted atoms, using indicator variables $\{d_k\}$ for the rounds in which the atoms occur, similar to [PZW+10]:

$$G = \sum_{k=1}^{\infty} E_k e^{-T_k} \delta_{\omega_k}, \tag{3.4}$$

where $E_k \overset{iid}{\sim} \text{Exp}(c)$, $T_k \overset{ind}{\sim} \text{Gamma}(d_k, \alpha)$, $\sum_{k=1}^{\infty} \mathbb{1}_{(d_k=r)} \overset{iid}{\sim} \text{Poisson}(\gamma)$, $\omega_k \overset{iid}{\sim} \frac{1}{\gamma} H_0$. We also place gamma priors on α, γ and c : $\alpha \sim \text{Gamma}(a_1, a_2), \gamma \sim \text{Gamma}(b_1, b_2)$, $c \sim \text{Gamma}(c_1, c_2)$. Denoting the data, the latent prior variables and the model hyperparameters by $\mathscr{D}, \Pi$ and Λ respectively, the full likelihood may be written as $P(\mathscr{D}, \Pi|\Lambda) = P(\mathscr{D}, \Pi_{-G}|\Pi_G, \Lambda) \cdot P(\Pi_G|\Lambda)$ where $P(\Pi_G|\Lambda) = P(\alpha) \cdot P(\gamma) \cdot P(c) \cdot P(\mathbf{d}|\gamma) \cdot \prod_{k=1}^{K} P(E_k|c) \cdot P(T_k|d_k, \alpha) \cdot \prod_{n=1}^{N} P(z_{nk}|E_k, T_k)$. We truncate the infinite gamma process to K atoms, and take N to be the total number of datapoints. Π_{-G} denotes the set of the latent variables excluding those from the Poisson-Gamma prior; for instance, in factor analysis for topic models, this contains the Dirichlet-distributed factor variables (or topics).

From the Poisson likelihood, we have $z_{nk}|E_k, T_k \sim \text{Poisson}(E_k e^{-T_k})$, independently for each n. The distributions of T_k and $\mathbf{d}$ involve the indicator functions on the round indicator

variables d_k:

$$P(T_k|d_k,\alpha) = \frac{\alpha^{\nu_k(0)}}{\prod_{r\geq 1}\Gamma(r)^{\mathbb{1}_{(d_k=r)}}} T_k^{\nu_k(1)} e^{-\alpha T_k},$$

where $\nu_k(s) = \sum_{r\geq 1}(r-s)\mathbb{1}_{(d_k=r)}$. We use the same weighting factors in our distribution on $\mathbf{d}$ as [PCB11]. See [PCB11] for a discussions on how to approximate these factors in the variational algorithm.

3.3.2 The Variational Prior Distribution

Mean-field variational inference involves minimizing the KL divergence between the model posterior, and a suitably constructed *variational* distribution which is used as a more tractable alternative to the actual posterior distribution. To that end, we propose a fully-factorized variational distribution on the Poisson-Gamma prior as follows:

$$Q = q(\alpha)\cdot q(\gamma)\cdot q(c)\cdot \prod_{k=1}^{K} q(E_k)\cdot q(T_k)\cdot q(d_k)\cdot \prod_{n=1}^{N} q(z_{nk}),$$

where $q(E_k) \sim \text{Gamma}(\acute{\xi}_k, \acute{\epsilon}_k)$, $q(T_k) \sim \text{Gamma}(\acute{\iota}_k, \acute{\upsilon}_k)$, $q(\alpha) \sim \text{Gamma}(\kappa_1, \kappa_2)$, $q(\gamma) \sim \text{Gamma}(\tau_1, \tau_2)$, $q(c) \sim \text{Gamma}(\rho_1, \rho_2)$, $q(z_{nk}) \sim \text{Poisson}(\lambda_{nk})$, with $q(d_k) \sim \text{Mult}(\varphi_k)$. Instead of working with the actual KL divergence between the full posterior and the factorized proxy distribution, variational inference maximizes what is canonically known as the *evidence lower bound* (ELBO), a function that is the same as the KL divergence up to a constant. In our case it may be written as $\mathscr{L} = \mathbb{E}_Q \log P(\mathscr{D}, \Pi|\Lambda) - \mathbb{E}_Q \log Q$. We omit the full representation here for brevity.

3.3.3 The Variational Parameter Updates

Since we are using exponential family variational distributions, we leverage the closed form variational updates for exponential families wherever we can, and perform gradient ascent on the ELBO for the parameters of those distributions which do not have closed form

updates. We list the updates on the distributions of the prior below. The closed-form updates for the hyperparameters in $q(E_k), q(\alpha), q(c)$ and $q(\gamma)$ are as follows:

$$\begin{aligned}
\xi_k' &= \sum_{n=1}^{N} \mathbb{E}_Q(z_{nk}) + 1, \quad \varepsilon_k' = \mathbb{E}(c) + N \times \mathbb{E}_Q\left[e^{-T_k}\right], \\
\kappa_1 &= \sum_{k=1}^{K} \sum_{r \geq 1} r\varphi_k(r) + a_1, \quad \kappa_2 = \sum_{k=1}^{K} \mathbb{E}_Q(T_k) + a_2, \\
\rho_1 &= c_1 + K, \quad \rho_2 = \sum_{k=1}^{K} \mathbb{E}_Q(E_k) + c_2, \\
\tau_1 &= b_1 + K, \quad \tau_2 = \sum_{r \geq 1} \left\{ 1 - \prod_{k=1}^{K} \sum_{\acute{r}=1}^{r-1} \varphi_k(\acute{r}) \right\} + b_2.
\end{aligned}$$

The updates for the multinomial probabilities in $q(d_k)$ are given by:

$$\begin{aligned}
\varphi_k(r) \propto \exp\{ & r\mathbb{E}_Q(\log\alpha) - \log\Gamma(r) + (r-1)\mathbb{E}_Q(\log T_k) - \zeta \cdot \sum_{i \neq k} \varphi_i(r) \\
& - \mathbb{E}_Q(\gamma) \sum_{j=2}^{r} \prod_{k' \neq k} \sum_{r'=1}^{j-1} \varphi_{k'}(r')\}.
\end{aligned}$$

The variational distribution $q(T_k)$ does not lend itself to closed-form analytical updates, so we perform gradient ascent on the evidence lower bound. The variational updates for $q(z_{nk})$ and for the variational distributions on the latent variables in Π_{-G} are model dependent, and require some approximations for the factor analysis case. See [RK15] for details.

3.4 Other Algorithms

Here we briefly describe the two primary competing algorithms we developed based on constructions of the Gamma process: a variational inference algorithm from the naïve construction, and a Markov chain Monte Carlo sampler based on our construction.

3.4.1 Naïve Variational Inference

We derive a variational inference algorithm from a simpler construction of the Gamma process, where we multiply the stick-breaking construction of the Dirichlet process by a Gamma random variable. The construction can be written as:

$$G = G_0 \sum_{i=1}^{\infty} V_i \prod_{j=1}^{i-1} (1 - V_j) \delta_{\omega_i},$$

where $G_0 \sim \text{Gamma}(\alpha, c), \quad V_i \overset{iid}{\sim} \text{Beta}(1, \alpha), \quad \omega_i \overset{iid}{\sim} H_0.$

We use an equivalent form of the construction that is similar to the one used above :

$$G = G_0 \sum_{k=1}^{\infty} V_k e^{-T_k} \delta_{\omega_k},$$

where $G_0 \sim \text{Gamma}(\alpha, c)$, $V_k \overset{iid}{\sim} \text{Beta}(1, \alpha)$, $T_k \overset{ind}{\sim} \text{Gamma}(k-1, \alpha), \quad \omega_i \overset{iid}{\sim} H_0.$

As before, we place gamma priors on α and c : $\alpha \sim \text{Gamma}(a_1, a_2), c \sim \text{Gamma}(c_1, c_2)$. The closed-form coordinate ascent updates for G_0, α and c and the gradient ascent updates for $\{V_k, T_k\}$ are detailed in the supplementary.

3.4.2 The MCMC Sampler

As a baseline, we also derive and compare the variational algorithm with a standard MCMC sampler for this model. We use the construction in (3.4) for sampling from the model. To avoid inferring the latent variables in all the atom weights of the Poisson-Gamma prior, we use Monte Carlo techniques to integrate them out, as in [PZW$^+$10]. This affects posterior inference for the indicators z_{nk}, the round indicators $\mathbf{d}$ and the hyperparameters c and α. The posterior distribution for γ is closed form, as are those for the likelihood latent variables in Π_{-G}. The complete updates are described in the supplementary.

3.5 Evaluation

In our experiments we focus on the problem of learning latent topics in document corpora. Given an observed set of counts of vocabulary words in a set of documents, represented by say a $V \times N$ count matrix, where V is the vocabulary size and N the number of documents, we aim to learn K latent factors and their vocabulary realizations using Poisson factor analysis. In particular, we model the observed corpus count matrix D as $D \sim \text{Poi}(\Phi\mathbf{I})$, where the $V \times K$ matrix Φ models the factor loadings, and the $K \times N$ matrix $\mathbf{I}$ models the actual factor counts in the documents.

We implemented and analyzed the performance of three variational algorithms corresponding to four different priors on $\mathbf{I}$: the Poisson-gamma process prior from this chapter (abbreviated hereafter as VGP), a Poisson-gamma prior using the naïve construction of the gamma process (VnGP), the Bernoulli-beta prior from [PCB11] (VBP) and the IBP prior from [DVMGT09] (VIBP), along with the MCMC sampler mentioned above (SGP). For the Bernoulli-beta priors we modeled $\mathbf{I}$ as $\mathbf{I} = W \circ Z$ as in [PCB11], where the nonparametric priors are put on Z and a vague Gamma prior is put on W. For the VGP and SGP models we set $\mathbf{I} = Z$. In addition, for all four algorithms, we put a symmetric Dirichlet$(\beta_1, \ldots, \beta_V)$ prior on the columns of Φ. We added corresponding variational distributions for the variables in the collection denoted as Π_{-G} above. We use held-out per-word test log-likelihoods and times required to update all variables in Π in each iteration as our comparison metrics, with 80% of the data used for training. We used the same likelihood metric as [ZC12], with the samples replaced by the expectations of the variational distributions.

Synthetic Data. As a warm-up, we consider the performances of VGP and SGP on some synthetic data generated from this model. We generate 200 weighted atoms from the gamma prior using the stick-breaking construction, and use the Poisson likelihood to generate 3000 values for each atom to yield the indicator matrix Z. We simulated a

vocabulary of 200 terms, generated a 200×200 factor-loading matrix Φ using symmetric Dirichlet priors, and then generated $D = \text{Poi}(\Phi Z)$. For the VGP and VnGP, we measure the test likelihood after every iteration and average the results across 10 random restarts. These measurements are plotted in fig.3.1a. As shown, VGP's measured heldout likelihood converges within 10 iterations. The SGP traceplot shows the first thirty heldout likelihoods measured after burn-in. Per-iteration times were 15 seconds and 2.36 minutes for VGP (with $K = 125$) and SGP respectively. The SGP learned K online, with values oscillating around 50. SNBP refers to the Poisson-Gamma mixture ("NB process") sampler from [ZC12]. Its traceplot shows the first 30 likelihoods measured after 1000 burn-in iterations. We see that it performed similarly to our algorithms, though slightly worse.

Real data. We used a similar framework to model the count data from the KOS[3], NIPS[4], Psychological Review (PsyRev)[5], and New York Times[3] corpora. The vocabulary sizes are 2566, 13649, 6906 and 100872 respectively, while the document counts are 1281, 1740, 3430 and 300000 respectively. For each dataset, we ran all three variational algorithms with 10 random restarts each, measuring the held-out log-likelihoods and per-iteration runtimes for different values of the truncation factor K. The learning rates for gradient ascent updates were kept on the order of 10^{-4} for both VGP and VBP, with 5 gradient steps per iteration. A representative subset of results is shown in figs.3.1b through 3.1f.

We used vague gamma priors on the hyperparameters α, γ and c in the variational algorithms, and improper (1) priors for the sampler. We found the test likelihoods to be independent of these initializations. The results for the variational algorithms were dependent on the Dirichlet prior β on Φ, as noted in fig.3.1b. We therefore used the learned test likelihood after 100 iterations as a heuristic to select β. We found the three

[3]https://archive.ics.uci.edu/ml/datasets/Bag+of+Words

[4]http://www.stats.ox.ac.uk/ teh/data.html

[5]http://psiexp.ss.uci.edu/research/programs_data/toolbox.htm

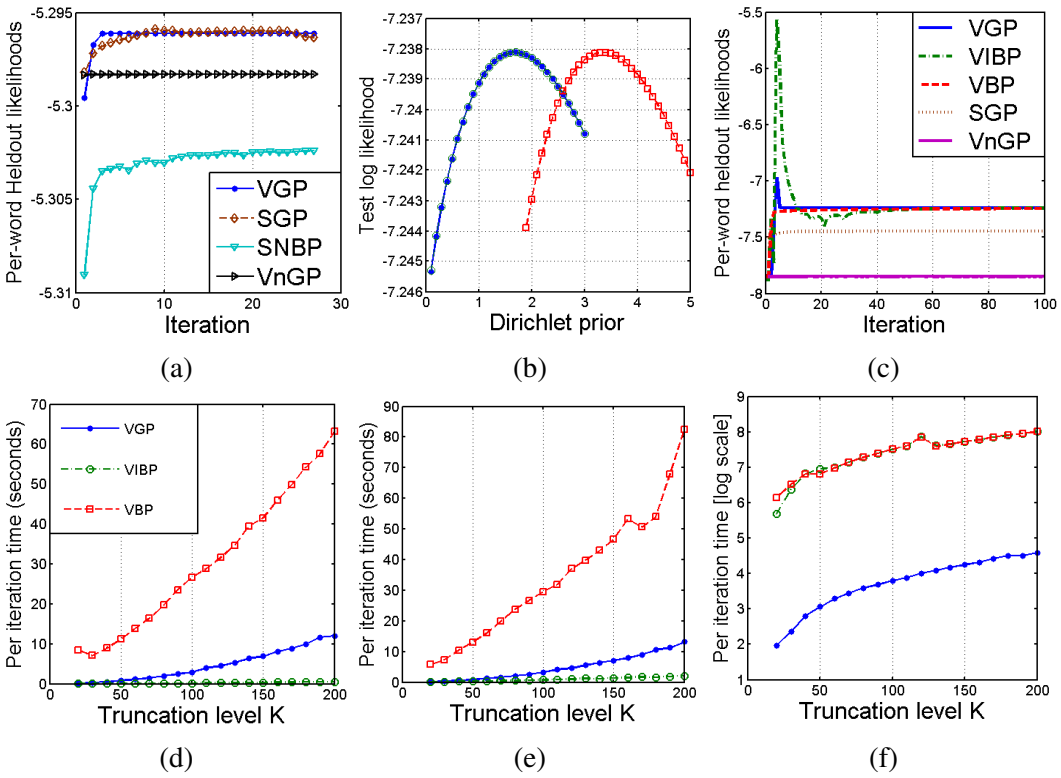


Figure 3.1: Plots of held-out test likelihoods and per-iteration running times (best viewed in color). Plots (d), (e) and (f) are for PsyRev, KOS, and NYT respectively. Plots (b) and (c) are for the PsyRev dataset. Algorithm trace colors are common to plots. See text for full details.

variational algorithms to attain very similar test likelihoods across all four datasets after a few hours of CPU time, with the VGP and VBP having a slight edge over the VIBP. The sampler somewhat unexpectedly did not attain a competitive score for any dataset, unlike the synthetic case. For instance, as shown in fig.3.1c, it oscillated around -7.45 for the PsyRev dataset, whereas the variational algorithms attained -7.23. For comparison, the NB process sampler from [ZC12] attains -7.25 each iteration after 1000 iterations of burn-in. VnGP was the worst performer, with stable log-likelihood of -7.85. Also as seen in fig.3.1c, VGP was faster to convergence (in less than 10 iterations in $\sim$5 seconds) than VIBP and VBP ($\sim$ 50 iterations each). The test log-likelihoods after a few hours of runtime were largely

independent of the truncation K for the three variational algorithms. Behavior for the other datasets was similar.

Among the three variational algorithms, the VIBP scaled best for small to medium datasets as a function of the truncation factor due to all updates being closed-form, in spite of having to learn the additional weight matrix W. The VGP running times were competitive for small values of K for these datasets. However, in the large NYT dataset, VGP was orders of magnitude faster than the Bernoulli-beta algorithms (note the log-scale in fig.3.1f). For example, with a truncation of 100 atoms, VGP took around 45 seconds per iteration, whereas both VIBP and VBP took more than 3 minutes. The VBP scaled poorly for all datasets, as seen in figs.3.1d through 3.1f. The reason for this is three-fold: learning the parameters for the additional matrix W which is directly affected by dimensionality (also the reason for VIBP being slow for NYT dataset), gradient updates for two variables (as opposed to one for VGP) and a Taylor approximation required for these gradient updates (see [PCB11]). The sampler SGP required around 7 minutes per iteration for the small datasets and an hour and 40 minutes on average for NYT.

To summarize, we found the VGP to post running times that are competitive with the fastest algorithm (VIBP) in small to medium datasets, and outperform the other methods completely in the large NYT dataset, all the while providing similar accuracy compared to the other variational algorithms, as measured by held-out likelihood. It was also the fastest to converge, typically taking less than 15 iterations. Compared with SGP, our variational method is substantially faster (particularly on large-scale data) and produces higher likelihood scores on real data.

3.6 Conclusion

We have described a novel stick-breaking representation for gamma processes and used it to derive a variational inference algorithm. This algorithm has been shown to be far more scalable for large datasets than related variational algorithms, while attaining similar accuracy and outperforming sampling-based methods. We expect that recent improvements to variational techniques can also be applied to our algorithm, potentially yielding even further scalability.

3.7 Proof Details

3.7.1 Variational inference details

To effectively perform variational inference, we re-write G as a single sum of weighted atoms, using indicator variables $\{d_k\}$ for the rounds in which the atoms occur, similar to [PZW+10]. We re-state our construction of the gamma CRM that we use for the inference algorithms:

$$G = \sum_{k=1}^{\infty} E_k e^{-T_k} \delta_{\omega_k}, \tag{3.5}$$

where $E_k \overset{iid}{\sim} \text{Exp}(c)$, $T_k \overset{ind}{\sim} \text{Gamma}(d_k, \alpha)$, $\sum_{k=1}^{\infty} \mathbb{1}_{(d_k=r)} \overset{iid}{\sim} \text{Poisson}(\gamma)$, $\omega_k \overset{iid}{\sim} \frac{1}{\gamma} H_0$. Here d_k denotes the round in which atom k appears, and may be defined as $d_k \overset{\Delta}{=} 1 + \sum_{i=1}^{\infty} \mathbb{I}\left\{ \sum_{j=1}^{i} C_j < k \right\}$. Conversely, given the round indicators $\mathbf{d} = \{d_k\}$, we can recover the round-specific atom counts as $C_i = \sum_{k=1}^{\infty} \mathbb{I}(d_k = i)$.

We place gamma priors on α, γ and c : $\alpha \sim \text{Gamma}(a_1, a_2), \gamma \sim \text{Gamma}(b_1, b_2), c \sim \text{Gamma}(c_1, c_2)$. Denoting the data, the latent prior variables and the model hyperparameters by $\mathscr{D}, \Pi$ and Λ respectively, the full likelihood may be written as $P(\mathscr{D}, \Pi | \Lambda) =$

$$P(\mathscr{D}, \Pi_{-G} | \Pi_G, \Lambda) \cdot P(\alpha) \cdot P(\gamma) \cdot P(c) \cdot P(\mathbf{d}|\gamma) \cdot \prod_{k=1}^{K} P(E_k|c) \cdot P(T_k|d_k, \alpha) \cdot \prod_{n=1}^{N} P(z_{nk}|E_k, T_k),$$

with Π_{-G} denoting the set of the latent variables excluding those from the Poisson-Gamma prior. The distribution of $\mathbf{d}$ is given by $P(\mathbf{d}|\gamma) =$

$$\prod_{r=1}^{\infty} \frac{\gamma^{\sum_k \mathbb{1}_{(d_k=r)}}}{\left(\sum_k \mathbb{1}_{(d_k=r)}\right)!} \cdot \exp\left\{ -\gamma \mathbb{I}\left(\sum_{r'=r}^{\infty} \sum_{k=1}^{\infty} \mathbb{1}_{(d_k=r')} > 0 \right) \right\}.$$

See [PCB11] for discussions on how to approximate some of these factors in the variational algorithm.

3.7.1.1 The Variational Prior Distribution

Mean-field variational inference involves minimizing the KL divergence between the model posterior, and a suitably constructed *variational* distribution which is used as a

more tractable alternative to the actual posterior distribution. To that end, we propose a fully-factorized variational distribution on the Poisson-Gamma prior as follows:

$$Q = q(\alpha) \cdot q(\gamma) \cdot q(c) \cdot \prod_{k=1}^{K} q(E_k) \cdot q(T_k) \cdot q(d_k) \cdot \prod_{n=1}^{N} q(z_{nk}),$$

where $q(E_k) \sim \text{Gamma}(\acute{\xi}_k, \acute{\varepsilon}_k)$, $q(T_k) \sim \text{Gamma}(\acute{\iota}_k, \acute{\upsilon}_k)$, $q(\alpha) \sim \text{Gamma}(\kappa_1, \kappa_2)$, $q(\gamma) \sim \text{Gamma}(\tau_1, \tau_2)$, $q(c) \sim \text{Gamma}(\rho_1, \rho_2)$, $q(z_{nk}) \sim \text{Poisson}(\lambda_{nk})$, while $q(d_k) \sim \text{Mult}(\varphi_k)$. The *evidence lower bound* (ELBO) may therefore be written as $\mathscr{L} = \mathbb{E}_Q \log P(\mathscr{D}, \Pi|\Lambda) - \mathbb{E}_Q \log Q$, with the relevant distributions described above.

3.7.1.2 Variational parameter updates

We first re-state the closed form updates for the variational distributions on the prior variables. The updates for the hyperparameters in $q(E_k), q(\alpha), q(c)$ and $q(\gamma)$ are as follows:

$$\begin{aligned}
\acute{\xi}_k &= \sum_{n=1}^{N} \mathbb{E}_Q(z_{nk}) + 1, \quad \acute{\varepsilon}_k = \mathbb{E}(c) + N \times \mathbb{E}_Q\left[e^{-T_k}\right], \\
\kappa_1 &= \sum_{k=1}^{K} \sum_{r \geq 1} r\varphi_k(r) + a_1, \quad \kappa_2 = \sum_{k=1}^{K} \mathbb{E}_Q(T_k) + a_2, \\
\rho_1 &= c_1 + K, \quad \rho_2 = \sum_{k=1}^{K} \mathbb{E}_Q(E_k) + c_2, \\
\tau_1 &= b_1 + K, \quad \tau_2 = \sum_{r \geq 1} \left\{ 1 - \prod_{k=1}^{K} \sum_{\acute{r}=1}^{r-1} \varphi_k(\acute{r}) \right\} + b_2.
\end{aligned}$$

The updates for the multinomial probabilities in $q(d_k)$ are given by:

$$\begin{aligned}
\varphi_k(r) \propto \exp\{ & r\mathbb{E}_Q(\log \alpha) - \log\Gamma(r) + (r-1)\mathbb{E}_Q(\log T_k) - \zeta \cdot \sum_{i \neq k} \varphi_i(r) \\
& - \mathbb{E}_Q(\gamma) \sum_{j=2}^{r} \prod_{k' \neq k} \sum_{r'=1}^{j-1} \varphi_{k'}(r')\}.
\end{aligned}$$

Next we describe the gradient ascent updates on $q(T_k)$ and the updates on $q(\Pi_{-G})$ and $q(z_{nk})$. The gradients for the two variational parameters in $q(T_k)$ are:

$$\begin{aligned}
\frac{\partial \mathscr{L}}{\partial \acute{u}_k} &= \sum_{r\geq 1}(r-1)\varphi_k(r)\psi^{'}(\acute{u}_k) - \frac{\mathbb{E}_Q(\alpha)}{\acute{v}_k} - \sum_{n=1}^{N}\mathbb{E}_Q(E_k)\left(\frac{\acute{v}_k}{\acute{v}_k+1}\right)^{\acute{u}_k}\cdot\log\left(\frac{\acute{v}_k}{\acute{v}_k+1}\right) \\
&\quad - \sum_{n=1}^{N}\mathbb{E}_Q(z_{nk})\frac{1}{\acute{v}_k} - (\acute{u}_k-1)\psi^{'}(\acute{u}_k) - 1, \\
\frac{\partial \mathscr{L}}{\partial \acute{v}_k} &= -\sum_{r\geq 1}(r-1)\varphi_k(r)\frac{1}{\acute{v}_k} + \mathbb{E}_Q(\alpha)\frac{\acute{u}_k}{(\acute{v}_k)^2} - \sum_{n=1}^{N}\mathbb{E}_Q(E_k)\acute{u}_k\frac{\acute{v}_k^{\acute{u}_k-1}}{(\acute{v}_k+1)^{\acute{u}_k+1}} \\
&\quad + \sum_{n=1}^{N}\mathbb{E}_Q(z_{nk})\frac{\acute{u}_k}{(\acute{v}_k)^2} - \frac{1}{\acute{v}_k}.
\end{aligned}$$

For the topic modeling problems, we model the observed vocabulary-vs-document corpus count matrix D as $D \sim \text{Poi}(\Phi Z)$, where the $V \times K$ matrix Φ models the factor loadings, and the $K \times N$ matrix Z models the actual factor counts in the documents. We put the $K-$truncated Poisson-Gamma prior on Z, and put a Dirichlet$(\beta_1,\ldots,\beta_V)$ prior on the columns of Φ.

The variational distribution Q consequently gets a Dirichlet$(\Phi|\{\mathbf{b}\}_k)$ distribution multiplied to it, where $\mathbf{b} = (b_1,\ldots,b_V)$ are the variational Dirichlet hyperparameters. This setup does not immediately lend itself to closed form updates for the b-s, so we resort to gradient ascent. The gradient of the ELBO with respect to each variational hyperparameter is

$$\begin{aligned}
\frac{\partial \mathscr{L}}{\partial b_{vk}} = -\mathbb{E}_Q(z_{nk})\cdot\frac{\sum_v b_{vk} - b_{vk}}{(\sum_v b_{vk})^2} + \psi^{'}(b_{vk})\cdot\left(\beta_v - b_{vk} + \sum_n d_{vn}\right) + \psi^{'}(\sum_v b_{vk})\times \\
\left(\sum_v b_{vk} - V - \beta_v - \sum_n d_{vn} + 1\right).
\end{aligned}$$

In practice however we found a closed-form update facilitated by a simple lower bound on the ELBO to converge faster. We describe the update here. First note that the part of the ELBO relevant to a potential closed form variational update of ϕ_{vk} can be written as

$$\mathscr{L} = -\phi_{vk}\cdot\sum_n \mathbb{E}_Q(z_{nk}) + \sum_n d_{vn}\cdot\log\phi_{vk} + \cdots,$$

which can then be lower bounded as

$$\mathscr{L} \geq \log \phi_{vk} \cdot \left(-\sum_n \mathbb{E}_Q(z_{nk}) + \sum_n d_{vn} \right) + \cdots .$$

This allows us to analytically update b_{vk} as $b_{vk} = -\sum_n \mathbb{E}_Q(z_{nk}) + \sum_n d_{vn} + \beta_v$. This frees us from having to choose appropriate corpus-specific initializations and learning rates for the Φs.

A similar lower bound on the ELBO allows us to update the variational parameters of $q(z_{nk})$ as $\lambda_{nk} = -1 - \sum_v d_{vn} + \mathbb{E}_Q(\log E_k) + \mathbb{E}_Q(T_k)$.

3.7.2 Variational inference using denormalized DP construction

We describe our algorithm derived from the simpler construction of the Gamma process by multiplying the stick-breaking construction of the Dirichlet process by a Gamma random variable. The construction can be written as:

$$G = G_0 \sum_{i=1}^{\infty} V_i \prod_{j=1}^{i-1} (1 - V_j) \delta_{\omega_i},$$

where $G_0 \sim \text{Gamma}(\alpha, c), \quad V_i \overset{iid}{\sim} \text{Beta}(1, \alpha), \quad \omega_i \overset{iid}{\sim} H_0.$

We use an equivalent form of the construction that is similar to the one used above :

$$G = G_0 \sum_{k=1}^{\infty} V_k e^{-T_k} \delta_{\omega_k},$$

where $G_0 \sim \text{Gamma}(\alpha, c), \quad V_k \overset{iid}{\sim} \text{Beta}(1, \alpha), \quad T_k \overset{ind}{\sim} \text{Gamma}(k-1, \alpha), \quad \omega_i \overset{iid}{\sim} H_0.$

As before, we place gamma priors on α and c : $\alpha \sim \text{Gamma}(a_1, a_2), c \sim \text{Gamma}(c_1, c_2)$.

Our variational distribution for this prior is as follows:

$$Q = q(G_0) \cdot q(\alpha) \cdot q(c) \cdot \prod_{k=1}^{K} q(V_k) \cdot q(T_k) \cdot \prod_{n=1}^{N} q(z_{nk}),$$

where $q(G_0) \sim \text{Gamma}(g_1, g_2), \quad q(V_k) \sim \text{Beta}(\nu_{k1}, \nu_{k2}), \quad q(T_k) \sim \text{Gamma}(t_{k1}, t_{k2}),$

$q(\alpha) \sim \text{Gamma}(\kappa_1, \kappa_2), \quad q(c) \sim \text{Gamma}(\rho_1, \rho_2), \quad q(z_{nk}) \sim \text{Poisson}(\lambda_{nk}).$

The closed form updates for the variational hyperparameters for $\alpha, G_0,$ and c are as follows:

$$\kappa_1 = a_1, \quad \kappa_2 = a_2 - \mathbb{E}_Q(\log G_0) - \sum_k \mathbb{E}_Q(\log(1-V_k) + \sum_k \mathbb{E}_Q(T_k),$$
$$g_1 = \alpha + \sum_{n=1}^{N}\sum_k \mathbb{E}_Q(z_{nk}), \quad g_2 = N \cdot \sum_k \mathbb{E}_Q(V_k e^{-T_k}),$$
$$\rho_1 = c_1, \quad \rho_2 = c_2 + \mathbb{E}_Q(G_0).$$

The updates for $q(V_k)$ and $q(T_k)$ are not closed form, necessitating gradient ascent steps; the gradients for the variational parameters in $q(V_k)$ are:

$$\frac{\partial \mathscr{L}}{\partial \nu_{k1}} = \psi'(\nu_{k1}+\nu_{k2})\left[\nu_{k1}+\nu_{k2}-\alpha-\sum_{n=1}^{N}\mathbb{E}_Q(z_{nk})-1\right] + \psi'(\nu_{k1})\cdot\left[\sum_{n=1}^{N}\mathbb{E}_Q(z_{nk})-\nu_{k1}+1\right] - N\cdot\mathbb{E}_Q\left(G_0 e^{-T_k}\right)\frac{\nu_{k2}}{\nu_{k1}+\nu_{k2}},$$
$$\frac{\partial \mathscr{L}}{\partial \nu_{k2}} = \psi'(\nu_{k1}+\nu_{k2})\left[\nu_{k1}+\nu_{k2}-\alpha-\sum_{n=1}^{N}\mathbb{E}_Q(z_{nk})-1\right] - N\cdot\mathbb{E}_Q\left(G_0 e^{-T_k}\right)\frac{\nu_{k1}}{\nu_{k1}+\nu_{k2}} + \psi'(\nu_{k2})\cdot[\alpha-\nu_{k2}].$$

The gradients for the variational parameters in $q(T_k)$ are:

$$\frac{\partial \mathscr{L}}{\partial t_{k1}} = 1 + \psi'(t_{k1})\cdot(k-t_{k1}-1) - \log t_{k2} - \frac{1}{t_{k2}}\left(\alpha+\sum_{n=1}^{N}\mathbb{E}_Q(z_{nk})\right) - N\cdot\mathbb{E}_Q(G_0V_k)\cdot\frac{\partial}{\partial t_{k1}}\left(\frac{t_{k2}}{t_{k2}+1}\right)^{t_{k1}},$$
$$\frac{\partial \mathscr{L}}{\partial t_{k2}} = \frac{t_{k1}}{t_{k2}^2}(\alpha+\sum_{n=1}^{N}\mathbb{E}_Q(z_{nk})) - \frac{1}{t_{k2}}(k-1) - N\cdot\mathbb{E}_Q(G_0V_k)\cdot\frac{\partial}{\partial t_{k2}}\left(\frac{t_{k2}}{t_{k2}+1}\right)^{t_{k1}}.$$

3.7.3 Markov chain Monte Carlo sampling details

We re-write the construction of the Poisson-Gamma prior:

$$G = \sum_{k=1}^{\infty} E_k e^{-T_k}\delta_{\omega_k},$$

$E_k \overset{iid}{\sim} \text{Exp}(c)$, $T_k \overset{ind}{\sim} \text{Gamma}(d_k, \alpha)$, $\sum_{k=1}^{\infty} \mathbb{1}_{(d_k=r)} \overset{iid}{\sim} \text{Pois}(\gamma)$, $\omega_k \overset{iid}{\sim} \frac{1}{\gamma} H_0$. We put improper priors on α and c, and a noninformative Gamma prior on γ. The indicator counts are given by $Z_{nk} \sim \text{Pois}(g_k)$, where $g_k = E_k e^{-T_k}$. To avoid sampling the atom weights E_k and T_k, we integrate them out using Monte Carlo techniques in the sampling steps for the prior.

3.7.3.1 Sampling the round indicators

The conditional posterior for the round indicators $\mathbf{d} = \{d_k\}_{k=1}^K$ can be written as

$$p\left(d_k = i | \{d_l\}_{l=1}^{k-1}, \{Z_{nk}\}_{n=1}^N, \alpha, c, \gamma\right)$$
$$\propto p\left(\{Z_{nk}\}_{n=1}^N | d_k = i, \alpha, c\right) p\left(d_k = i | \{d_l\}_{l=1}^{k-1}\right).$$

For the first factor, we collapse out the stick-breaking weights and approximate the resulting integral using Monte-Carlo techniques as follows:

$$p\left(\{Z_{nk}\}_{n=1}^N | d_k = i, \alpha, c\right) = \int_{[0,\infty]^i} \prod_{n=1}^N \text{Pois}(Z_{nk}|g_k) \text{d}G$$
$$\approx \frac{1}{S} \sum_{s=1}^S \prod_{n=1}^N \text{Pois}(Z_{nk}|g_k^{(s)}),$$

where $g_k^{(s)} = E_k^{(s)} e^{-T_k^{(s)}} \overset{d}{=} V_{k,d_k}^{(s)} \prod_{l=1}^{d_k} (1 - V_{kl}^{(s)})$. Here S is the number of simulated samples from the integral over the stick-breaking weights. We take $S = 1000$ in our experiments.

The second factor is the same as [PZW+10]:

$p(d_k = d | \gamma, \{d_l\}_{l=1}^{k-1}) =$

$$\begin{cases} 0 & \text{if } d < d_{k-1} \\ \frac{1 - \sum_{t=1}^{D_{k-1}} \text{Pois}(t|\gamma)}{1 - \sum_{t=1}^{D_{k-1}-1} \text{Pois}(t|\gamma)} & \text{if } d = d_{k-1} \\ \left(1 - \frac{1 - \sum_{t=1}^{D_{k-1}} \text{Pois}(t|\gamma)}{1 - \sum_{t=1}^{D_{k-1}-1} \text{Pois}(t|\gamma)}\right) (1 - \text{Pois}(0|\gamma)) \text{Pois}(0|\gamma)^{h-1} & \text{if } d = d_{k-1} + h \end{cases}$$

Here $D_k \overset{\Delta}{=} \sum_{j=1}^k \mathbb{I}(d_j = d_k)$. Normalizing the product of these two factors over all i is infeasible, so we evaluate this product for increasing i till it drops below 10^{-2}, and normalize over the gathered values.

3.7.3.2 Sampling the factor variables

Here we consider the Poisson factor modeling scenario that we use to model vocabulary-document count matrices. Recall that a $V \times N$ count matrix D is modeled as $D = \text{Poi}(\Phi Z)$, where the $V \times K$ matrix Φ models the factor loadings, and the $K \times N$ matrix Z models the actual factor counts in the documents.. We put the Poisson-Gamma prior on Z and symmetric Dirichlet$(\beta_1, \ldots, \beta_V)$ priors on the columns of Φ. The sampling steps for Φ and Z are described next. **Sampling Φ.**

First note that the elements of the count matrix are modeled as $d_{vn} = \text{Poi}\left(\sum_{k=1}^{K} \phi_{vk} z_{kn}\right)$, which can be equivalently written as $d_{vn} = \sum_{k=1}^{K} d_{vkn}, \quad d_{vkn} = \text{Poi}(\phi_{vk} z_{kn})$. Standard manipulations then allow us to sample the d_{vkn}'s from Mult$(d_{vn}; p_{v1n}, \ldots, p_{vKn})$ where $p_{vkn} = \phi_{vk} z_{kn} / \sum_{k}^{K} \phi_{vk} z_{kn}$.

Now we have $\phi_k \sim \text{Dirichlet}(\beta_1, \ldots, \beta_V)$. Using standard relationships between Poisson and multinomial distributions, we can derive the posterior distribution of the ϕ_k's as Dirichlet$(\beta_1 + d_{1k}, \ldots, \beta_V + d_{Vk})$, where $d_{vk} = \sum_{n=1}^{N} d_{vkn}$. **Sampling Z.**

In our algorithm we sample each z_{nk} conditioned on all the other variables in the model; therefore the conditional posterior distribution can be written as

$$
\begin{aligned}
& p(z_{nk} | D, \Phi, Z_{n,-k}, \mathbf{d}, \alpha, c, \gamma) \\
& = p(D | Z_n, \Phi) p(z_{nk} | \mathbf{d}, \alpha, c, Z_{n,-k}) \\
& = \prod_{v=1}^{V} \text{Poi}\left(d_{vn} | \sum_{k=1}^{K} \phi_{vk} z_{kn} \right) \frac{p(Z_n | \mathbf{d}, \alpha, c)}{p(Z_{n,-k} | \mathbf{d}, \alpha, c)}.
\end{aligned}
$$

The distributions in both the numerator and denominator of the second factor can be sampled from using the Monte Carlo techniques described above, by integrating out the stick-breaking weights.

3.7.3.3 Sampling hyperparameters

As mentioned above, we put a noninformative Gamma prior on γ and improper (1) priors on α and c. The posterior sampling steps are described below: **Sampling γ.**

Given the round indicators $\mathbf{d} = \{d_k\}$, we can recover the round-specific atom counts as described above. Then the conjugacy between the Gamma prior on γ and the Poisson distribution of C_i gives us a closed form posterior distribution for γ: $p(\gamma|\mathbf{d}, Z, \alpha, c) = \text{Gamma}(\gamma|a + \sum_{i=1}^{K} C_i, b + d_K)$. **Sampling α.**

The conditional posterior distribution of α may be written as:

$$p(\alpha|Z, \mathbf{d}, c) \propto p(\alpha) \prod_{n=1}^{N} \prod_{k=1}^{K} p(Z|\mathbf{d}, \alpha, c).$$

We calculate the posterior distribution of Z using Monte Carlo techniques as described above. Then we discretize the search space for α around its current values as $(\alpha_{cur} + t\Delta\alpha)_{t=L}^{U}$, where the lower and upper bounds L and U are chosen so that the unnormalized posterior falls below 10^{-2}. The search space is also clipped below at 0. α is then drawn from a multinomial distribution on the search values after normalization. **Sampling c.**

We sample c in exactly the same way as α. We first write the conditional posterior as

$$p(c|Z, \mathbf{d}, \alpha) \propto p(c) \prod_{n=1}^{N} \prod_{k=1}^{K} p(Z|\mathbf{d}, \alpha, c).$$

The search space $(c > 0)$ is then discretized using appropriate upper and lower bounds as above, and Z is sampled using Monte Carlo techniques. c is then drawn from a multinomial distribution on the search values after normalization.

Chapter 4: Robust Monte Carlo Sampling using Riemannian Nosé-Poincaré Hamiltonian Dynamics

Bayesian inference in high dimensional models often requires one to draw samples from posterior distributions of variables which cannot be computed in closed form. Monte Carlo techniques are the primary tool in one's arsenal for this purpose; they allow one to draw samples from a sequence of probability distributions that form a Markov chain with the target distribution as its stationary distribution. The Hamiltonian Monte Carlo (HMC) technique, first proposed in [DKPR87] as "Hybrid Monte Carlo", improves on this by using ideas from statistical physics to avoid the random walk behavior that normally arises in these Markov chains. HMC sets the target distribution as the "potential energy" of the simulated system, and uses auxiliary "momentum" variables to augment the potential with a kinetic energy term. Hamiltonian dynamics are then used to create a sampler that conserves this quantity, and the samples generated by this technique are provably less correlated among themselves which leads to faster convergence to the target distribution. The dynamics are usually specified with a set of differential equations which then have to be discretized; however one can derive discrete-time numeric integrators [Nea10, LR05], usually called "leapfrog" methods, that preserve the detailed balance and time reversibility properties of the continuous-time formulations.

In the statistical physics literature, dynamics-based techniques are used to sample from a canonical ensemble, where the possible states of the system remain at a constant temperature.

One technique that has been used for this purpose uses the Nosé Hamiltonian [Nos84], which generates sequence of states from the canonical ensemble under the standard ergodicity assumptions. However, as noted in [BLL99], one has to rescale the time variable in this system in order to derive evenly spaced out samples. The Nosé-Hoover system [Hoo85] proposes a change of variables that allows one to derive evenly spaced samples from the canonical ensemble, however the resulting system is not Hamiltonian. In particular, the numeric integrator one uses to solve the system of differential equations for this formulation is not symplectic, in that it does not preserve the symplectic geometry of the original manifold defined by the Hamiltonian system. Symplecticness being in general a stronger property than volume preservation, [BLL99] proposed a Hamiltonian which can be used to derive a numeric integrator that samples from a fixed temperature canonical ensemble, and is also symplectic and time-reversible.

Although Monte Carlo techniques constructed from these formulations can be used to sample from Markov chains that converge to the desired target distributions, a difficulty arises when working with very large-scale datasets. Here, due to computational limitations, one often cannot compute the gradient of the target likelihood (potential energy) over the entire dataset in a reasonable amount of time. Instead, at every iteration one computes a stochastic gradient, which is the gradient evaluated over a randomly selected "mini-batch" of data [RM51, WT11]. This allows the algorithms to scale to massive datasets commonly seen in machine learning, while preserving the desired theoretical properties. Recent work in this vein includes stochastic gradient Langevin dynamics [WT11], stochastic gradient Hamiltonian Monte Carlo [CCG14], and other variants and extensions [DFB$^+$14, PT13]. First order Langevin dynamics methods are inherently random-walk based, whereas the HMC methods exploit the exploration efficiencies of Hamiltonian systems to derive more robust samplers in a stochastic setting. However, the samples are not automatically drawn

from the canonical ensemble, an issue that was addressed in [DFB+14], where the SGNHT algorithm was proposed. The authors do show the efficacy of the sampler in the face of stochastic noise; but as mentioned above, the Nosé-Hoover system is not Hamiltonian, which can have adverse effects on the exploration efficiency and convergence speed of any MCMC algorithm derived using its dynamics.

Therefore one would want a technique that samples from the canonical ensemble, without sacrificing the advantages of Hamiltonian trajectories. Furthermore, one would also want a stochastic version that would scale to large datasets. To that end, we propose the stochastic gradient Nosé-Poincare Hamiltonian Monte Carlo sampler, which uses a variant of the Hamiltonian proposed in [BLL99] that leverages Riemannian preconditioning and corrects for the random noise from the stochastic gradients while preserving the desired properties mentioned above. The basic idea of Riemannian adaptations, first proposed in [GC11], is to define a Riemannian metric tensor on the parameter space and use structural cues from the resulting manifold while traversing the Hamiltonian trajectories. This technique was exploited in the context of sampling from a high dimensional probability simplex [PT13], where the geometric information allows the sampler to improve upon the slow mixing behavior exhibited by the first order Langevin dynamics on parameter spaces with a high degree of correlation. Another advantage of locally-adaptive preconditioning is that one does not have to worry about selecting optimal values for the "mass" matrices in Hamiltonian samplers, an aspect such samplers tend to be highly sensitive to [GC11, BLL99]. In our algorithm, we use Riemann tensors on the original parameter space to precondition the momenta of both the real and extended position variables in the Nosé Hamiltonian, and show that the resulting system samples from the canonical ensemble. We then add correction terms to account for noise when the full gradient is replaced by the stochastic gradient, and use the Fokker-Planck equation to prove that the dynamics conserve the desired energy.

Finally, we apply our algorithm to parameter estimation in synthetic settings and a high dimensional topic modeling scenario using the Poisson factor analysis framework [ZC15] and the exact Gamma process construction of [RK15].

4.1 Preliminaries

4.1.1 Monte Carlo using Hamiltonian Dynamics

Let us denote the model parameters by $\boldsymbol{\theta}$. Suppose we want to generate samples from the posterior distribution of $\boldsymbol{\theta}$ given data $\mathbf{X}$, $p(\boldsymbol{\theta}|\mathbf{X})$. In a Hamiltonian setting, we take the joint log likelihood of the data and the parameters, $\mathscr{L}(\boldsymbol{\theta}) = \log p(\mathbf{X}|\boldsymbol{\theta}) + \log p(\boldsymbol{\theta})$, and add to it a term involving auxiliary "momentum" variables $\mathbf{p}$, to get the Hamiltonian

$$H(\boldsymbol{\theta}, \mathbf{p}) = -\mathscr{L}(\boldsymbol{\theta}) + \frac{1}{2}\mathbf{p}^T \mathrm{M}^{-1}\mathbf{p}. \tag{4.1}$$

The quantity can be interpreted in a physical sense as the sum of the potential energy $\mathscr{L}(\boldsymbol{\theta})$ and the kinetic energy $\frac{1}{2}\mathbf{p}^T\mathrm{M}^{-1}\mathbf{p}$, where M^{-1} acts as the canonical mass matrix. The joint distribution of $\boldsymbol{\theta}$ and $\mathbf{p}$ is then defined as $p(\boldsymbol{\theta}, \mathbf{p}) \propto \exp(-H(\boldsymbol{\theta}, \mathbf{p}))$. It is easily seen that we can integrate out $\mathbf{p}$ from $p(\boldsymbol{\theta}, \mathbf{p})$ to get the desired posterior distribution on $\boldsymbol{\theta}$.

The dynamics of this system are governed by Hamilton's equations of motion

$$\dot{\theta} = \frac{\partial}{\partial \mathbf{p}} H(\boldsymbol{\theta}, \mathbf{p}), \quad \dot{\mathbf{p}} = -\frac{\partial}{\partial \boldsymbol{\theta}} H(\boldsymbol{\theta}, \mathbf{p}),$$

where we have denoted the time derivatives with the dot accent, i.e. $\dot{\theta} = d\boldsymbol{\theta}/dt$. In our formulation the dynamics are

$$\dot{\theta} = \mathrm{M}^{-1}\mathbf{p}, \quad \dot{\mathbf{p}} = \nabla \mathscr{L}(\boldsymbol{\theta}).$$

These equations are time-reversible, and the dynamics conserve the total energy and are symplectic as well. These continuous-time equations are discretized to give "leapfrog" algorithms which are used for Monte Carlo simulations along with Metropolis-Hastings correction steps. For details see [Nea10, LR05].

4.1.2 Riemann Adjusted Hamiltonian Monte Carlo

The scaling issues associated with standard Hamiltonian Monte Carlo algorithms can be partially alleviated using Riemannian preconditioning. We first define the Hamiltonian on a Riemann manifold defined by a positive definite metric $\mathbf{G}(\boldsymbol{\theta})$. Trajectories incorporating information from this manifold can be simulated by simply defining the kinetic energy in terms of the metric tensor [GC11], which leads to the following Hamiltonian:

$$H_{gc}(\boldsymbol{\theta},\mathbf{p}) = -\mathscr{L}(\boldsymbol{\theta}) + \frac{1}{2}\mathbf{p}^T\mathbf{G}(\boldsymbol{\theta})^{-1}\mathbf{p} + \frac{1}{2}\log\left\{(2\pi)^D|\mathbf{G}(\boldsymbol{\theta})|\right\} \tag{4.2}$$

where D is the dimensionality of the parameter space. The log term ensures that the momentum variable $\mathbf{p}$ can be integrated out to recover the desired marginal density of $\boldsymbol{\theta}$. The equations of motion in this system therefore are

$$\dot{\theta} = \mathbf{G}(\boldsymbol{\theta})^{-1}\mathbf{p}, \quad \dot{\mathbf{p}} = \nabla\mathscr{L}(\boldsymbol{\theta}) - \frac{1}{2}\mathrm{tr}\left(\mathbf{G}(\boldsymbol{\theta})^{-1}\nabla\mathbf{G}(\boldsymbol{\theta})\right) + \frac{1}{2}\mathbf{p}^T\mathbf{G}(\boldsymbol{\theta})^{-1}\nabla\mathbf{G}(\boldsymbol{\theta})\mathbf{G}(\boldsymbol{\theta})^{-1}\mathbf{p}.$$

To discretize this system of equations, we use the generalized leapfrog algorithm, where a first order symplectic integrator is composed with its adjoint; the resultant second order integrator can be shown to be both time-reversible and symplectic [LR05]. We use this integrator to derive discretized samplers from our Nosé-Poincaré Hamiltonians in §6.2.

4.1.3 Stochastic Gradient Dynamics

For moderate datasets, the gradient of the log-likelihood in the dynamics above can be evaluated over the entire dataset. However, for large datasets, doing so in every iteration becomes prohibitively expensive. The most common way around this problem is to replace the full gradient by one evaluated over a random "mini-batch" of the dataset, a technique inspired by [RM51]. The approximate gradient of the log-likelihood can be written as

$$\nabla\tilde{\mathscr{L}}(\boldsymbol{\theta}) = \frac{|N|}{|\tilde{N}|}\sum_{x\in\tilde{N}}\nabla\log p(x|\boldsymbol{\theta}) + \nabla\log p(\boldsymbol{\theta}),$$

where N denotes the entire dataset, and $\tilde{N}$ denotes a random mini-batch. Monte Carlo samplers using stochastic gradients have been proposed for first order Langevin dynamics [WT11, PT13], as well as for Hamiltonian systems using second order Langevin dynamics [CCG14].

Another feature of these algorithms is the removal of a Metropolis-Hastings correction step, as that would require very expensive computations over the entire dataset. Instead, a decaying sequence of stepsizes $\{\varepsilon_t\}$ satisfying $\sum_t \varepsilon_t = \infty$ and $\sum_t \varepsilon_t^2 < \infty$ is used, for which the Markov chain of distributions can be proved to have the desired target as its equilibrium distribution.

4.2 Riemannian Nosé-Poincaré Dynamics

4.2.1 The Deterministic Case

The Nosé-Poincaré Hamiltonian proposed in [BLL99] can be written as

$$H(\boldsymbol{\theta}, \mathbf{p}, s, q) = s\left(-\mathscr{L}(\boldsymbol{\theta}) + \frac{1}{2}\left(\frac{\mathbf{p}}{s}\right)^T \mathrm{M}^{-1}\frac{\mathbf{p}}{s} + \frac{q^2}{2Q} + gkT\log s - H_0\right). \tag{4.3}$$

Here k is Boltzmann's constant, T is the system temperature, g is equal to the number of degrees of freedom of the system. s is an extended position variable with momentum q and associated mass Q, as introduced by [Nos84]. s acts as the time-scaling function in a Poincaré transformation, which allows us to preserve the dynamics of the original Hamiltonian upto the time transformation.

4.2.1.1 The Riemann augmentation

As mentioned previously, the dynamics of Hamiltonian systems are highly sensitive to the values of the mass matrices, in this case M and Q. To take advantage of locally-adaptive walks on the Riemann manifold, we replace the mass matrices in (4.3) by a metric tensor

$\mathbf{G}(\boldsymbol{\theta})$ as follows:

$$H(\boldsymbol{\theta},\mathbf{p},s,q) = s\Bigg(-\mathscr{L}(\boldsymbol{\theta}) + \frac{1}{2}\left(\frac{\mathbf{p}}{s}\right)^T \mathbf{G}(\boldsymbol{\theta})^{-1}\frac{\mathbf{p}}{s} + \frac{1}{2}|\mathbf{G}(\boldsymbol{\theta})|^{-1}q^2 + \frac{1+kT}{2}\log\left\{(2\pi)^D|\mathbf{G}(\boldsymbol{\theta})|\right\} + gkT\log s - H_0\Bigg). \tag{4.4}$$

The log term ensures that we can integrate out the extended momentum term q to get back the Hamiltonian (6.2), or for that matter the one in (6.1), if one treats the curvature of the metric-defined manifold as a constant. Note that the only constant one needs to choose in this system is the value for T. In Bayesian inference we usually take $kT = 1$, though use of this Hamiltonian is not restricted to that specific choice.

Theorem 1. *The Riemannian Nosé-Poincaré Hamiltonian defined in* (4.4) *generates samples from the canonical ensemble.*

Proof. We will show that we can integrate out s, q from $p(\boldsymbol{\theta},\mathbf{p},s,q) \propto \exp\left(-H(\boldsymbol{\theta},\mathbf{p},s,q)\right)$ to get $p(\boldsymbol{\theta},\mathbf{p}) \propto \exp\left(-H_{gc}(\boldsymbol{\theta},\mathbf{p})/kT\right)$. The integration of s essentially follows [BLL99], so we detail that in §A of the supplementary. With s integrated out, we are left with $p(\boldsymbol{\theta},\mathbf{p},q) \propto \exp\left(-H_{gc}(\boldsymbol{\theta},\mathbf{p})/kT\right)\exp\left[-\frac{1}{2kT}|\mathbf{G}(\boldsymbol{\theta})|^{-1}q^2 - \frac{1}{2}\log\left\{(2\pi)^D|\mathbf{G}(\boldsymbol{\theta})|\right\}\right]$. We can easily integrate out q from the second exponential term on the right to get the desired form for $p(\boldsymbol{\theta},\mathbf{p})$. □

The dynamics for this system are given by

$$\begin{aligned}
\dot{\theta} &= \left(\frac{\mathbf{p}}{s}\right)^T \mathbf{G}(\boldsymbol{\theta})^{-1} \\
\dot{\mathbf{p}} &= s\frac{1}{2}\left(\frac{\mathbf{p}}{s}\right)^T \mathbf{G}(\boldsymbol{\theta})^{-1}\nabla\mathbf{G}(\boldsymbol{\theta})\mathbf{G}(\boldsymbol{\theta})^{-1}\left(\frac{\mathbf{p}}{s}\right) + s\frac{1}{2}q^2|\mathbf{G}(\boldsymbol{\theta})|^{-1}\mathrm{tr}\left(\mathbf{G}(\boldsymbol{\theta})^{-1}\nabla\mathbf{G}(\boldsymbol{\theta})\right) \\
&\quad - \frac{s}{2}(1+kT)\mathrm{tr}\left(\mathbf{G}(\boldsymbol{\theta})^{-1}\nabla\mathbf{G}(\boldsymbol{\theta})\right) + s\nabla\mathscr{L}(\boldsymbol{\theta}) \\
\dot{s} &= sq|\mathbf{G}(\boldsymbol{\theta})|^{-1}, \quad \dot{q} = -gkT + \left(\frac{\mathbf{p}}{s}\right)^T \mathbf{G}(\boldsymbol{\theta})^{-1}\frac{\mathbf{p}}{s} - H_{\text{inner}}
\end{aligned} \tag{4.5}$$

where $H_{\text{inner}} = \Bigg(-\mathscr{L}(\boldsymbol{\theta}) + \frac{1}{2}\left(\frac{\mathbf{p}}{s}\right)^T \mathbf{G}(\boldsymbol{\theta})^{-1}\frac{\mathbf{p}}{s} + \frac{1}{2}|\mathbf{G}(\boldsymbol{\theta})|^{-1}q^2 + \frac{1+kT}{2}\log\left\{(2\pi)^D|\mathbf{G}(\boldsymbol{\theta})|\right\} + gkT\log s - H_0\Bigg)$.

Since we derived these equations following Hamiltonian's laws of motion, the dynamics are time-reversible and symplectic. Moreover, as shown above, the samples are drawn from the fixed temperature canonical ensemble, and the use of Riemann metric tensors allows us to exploit the geometry of the manifold. We can see that if the manifold is assumed to have a constant curvature. i.e. $\nabla \mathbf{G}(\boldsymbol{\theta}) = 0$, then the dynamics above reduce to those of the standard Nosé-Poincaré system in [BLL99].

We can also see that the Hamiltonian (4.4) is not a simple generalization of (6.2); the only way to recover (6.2) is to set the extended position variable $s = 1$, $q = 0$ (effectively assuming that the particles are not changing position in the extended phase space), as well as setting $T = 0$. This is problematic in a physical sense, since particles do not move at absolute zero.

4.2.1.2 The Discretized Dynamics

For Monte Carlo sampling we need to discretize this system, and to do so we use the generalized leapfrog algorithm of [LR05]. The generalized leapfrog algorithm can be shown to be both time-reversible and symplectic. However, since our Hamiltonian is not separable due to the coupling of the momenta terms with the position variable $\boldsymbol{\theta}$, the leapfrog equations are implicitly defined, necessitating the use of fixed point techniques or Newton-like iterations to solve them. Let us denote $\nabla_{\theta_i} H(\boldsymbol{\theta}, s, \mathbf{p}, q)$

$$= \frac{\varepsilon}{2}\Bigg[-\frac{1}{2} s \left(\frac{\mathbf{p}}{s}\right)^T G(\boldsymbol{\theta})^{-1} \left(\frac{\partial}{\partial \theta_i} G(\boldsymbol{\theta})\right) G(\boldsymbol{\theta})^{-1} \left(\frac{\mathbf{p}}{s}\right) + \frac{s}{2}(1+kT)\mathrm{tr}\left\{ G(\boldsymbol{\theta})^{-1} \frac{\partial}{\partial \theta_i} G(\boldsymbol{\theta}) \right\}$$
$$- s\frac{\partial}{\partial \theta_i}\mathscr{L}(\boldsymbol{\theta}) - s\frac{q^2}{2}|G(\boldsymbol{\theta})|^{-1}\mathrm{tr}\left\{ G(\boldsymbol{\theta})^{-1}\frac{\partial}{\partial \theta_i} G(\boldsymbol{\theta}) \right\} \Bigg].$$

Then, can apply this algorithm to the dynamics (4.5) to get the following discrete update equations:

$$
\begin{aligned}
p_i^{(t+\varepsilon/2)} &= p_i^{(t)} - \frac{\varepsilon}{2}\nabla_{\theta_i} H\left(\boldsymbol{\theta}^{(t)}, s^{(t)}, \mathbf{p}^{(t+\varepsilon/2)}, q^{(t+\varepsilon/2)}\right) \\
q^{(t+\varepsilon/2)} &= q^{(t)} - \frac{\varepsilon}{2}\left[H_{\text{inner}} + gkT - \left(\frac{\mathbf{p}^{(t+\varepsilon/2)}}{s^{(t)}}\right)^T G(\boldsymbol{\theta}^{(t)})^{-1}\left(\frac{\mathbf{p}^{(t+\varepsilon/2)}}{s^{(t)}}\right)\right] \\
\theta_i^{(t+\varepsilon} &= \theta_i^{(t)} + \frac{\varepsilon}{2}\left[\left(\frac{\mathbf{p}^{(t+\varepsilon/2)}}{s^{(t)}}\right)^T G(\boldsymbol{\theta}^{(t)})^{-1} + \left(\frac{\mathbf{p}^{(t+\varepsilon/2)}}{s^{(t+\varepsilon)}}\right)^T G(\boldsymbol{\theta}^{(t+\varepsilon)})^{-1}\right]_i \\
s^{(t+\varepsilon)} &= s^{(t)} + \frac{\varepsilon}{2}\left[s^{(t)} q^{(t+\varepsilon/2)} |G(\boldsymbol{\theta}^{(t)})|^{-1} + s^{(t+\varepsilon)} q^{(t+\varepsilon/2)} |G(\boldsymbol{\theta}^{(t+\varepsilon)})|^{-1}\right] \\
p_i^{(t+\varepsilon)} &= p_i^{(t+\varepsilon/2)} - \frac{\varepsilon}{2}\nabla_{\theta} H\left(\boldsymbol{\theta}^{(t+\varepsilon)}, s^{(t+\varepsilon)}, \mathbf{p}^{(t+\varepsilon/2)}, q^{(t+\varepsilon/2)}\right) \\
q^{(t+\varepsilon)} &= q^{(t+\varepsilon/2)} - \frac{\varepsilon}{2}\left[H_{\text{inner}} + gkT - \left(\frac{\mathbf{p}^{(t+\varepsilon/2)}}{s^{(t+\varepsilon)}}\right)^T G(\boldsymbol{\theta}^{(t+\varepsilon)})^{-1}\left(\frac{\mathbf{p}^{(t+\varepsilon/2)}}{s^{(t+\varepsilon)}}\right)\right].
\end{aligned}
\tag{4.6}
$$

The half-step $(t+\varepsilon/2)$ updates equations for the momenta $\mathbf{p}$ and q are implicitly defined, as are those for the $\boldsymbol{\theta}, s$ pair. The full step $(t+\varepsilon)$ updates for $\mathbf{p}$ and q are explicit, since they depend only on the half-step values of $\mathbf{p}$ and q and the full step ones for $\boldsymbol{\theta}, s$. The overall procedure is outlined in Algorithm 1.

In [GC11] the authors mention using fixed point iterations for solving a similar set of equations. However, in our experiments with real datasets, fixed point updates led to unstable mixing at even moderate learning rates. One reason for this could be the fact that the Jacobians of the implicit equations are large (implying "stiff" domains) for these datasets and this specific formulation. Therefore, for strictly positive parameters we resort to using diagonal metric tensors, and using Newton's method for solving the implicit systems. For instance, using the tensor $G(\boldsymbol{\theta}) = \text{diag}(\boldsymbol{\theta})^{-1}$, (assuming $G(\boldsymbol{\theta}) \succ 0$), the implicit system for $\mathbf{p}^{t+\varepsilon/2}$ and $q^{t+\varepsilon/2}$ reduces to the following quadratic system:

$$
\frac{\varepsilon}{4}\frac{p_i^2}{s^{(t)}} + p_i - p_i^{(t)} - \frac{\varepsilon}{2}s^{(t)}\left[\frac{1}{\theta_i^{(t)}}\left(1 + kT - |G(\boldsymbol{\theta}^{(t)})|^{-1}\frac{q^2}{2}\right) + \frac{\partial}{\partial \theta_i^{(t)}}\mathscr{L}(\boldsymbol{\theta}^{(t)})\right] = 0
$$

Algorithm 1 Riemann Nosé-Poincaré HMC

Input: $\boldsymbol{\theta}, \varepsilon, kT$
Initialize $\boldsymbol{\theta}, s$
repeat
· Sample $\mathbf{p}^{(t)} \sim N(0, G(\boldsymbol{\theta})), q^{(t)} \sim N(0, |G(\boldsymbol{\theta})|)$
·Perform leapfrog dynamics (6.14)
to get $((\boldsymbol{\theta}^{(t)}, s^{(t)}, \mathbf{p}^{(t)}, q^{(t)})$:
for $i = 1$ **to** $leapfrog\_iterations$ **do**
· Perform implicit Newton updates to get
$\mathbf{p}^{(t+\varepsilon/2)}, q^{(t+\varepsilon/2)}, \boldsymbol{\theta}^{(t+\varepsilon)}, q^{(t+\varepsilon)}$
· Perform explicit updates to get $\mathbf{p}^{(t+\varepsilon)}, q^{(t+\varepsilon)}$
end for
$\cdot(\boldsymbol{\theta}^{'}, s^{'}, \mathbf{p}^{'}, q^{'}) \leftarrow ((\boldsymbol{\theta}^{(t+\varepsilon)}, s^{(t+\varepsilon)}, \mathbf{p}^{(t+\varepsilon)}, q^{(t+\varepsilon)})$
· Set $((\boldsymbol{\theta}^{(t+1)}, s^{(t+1)}, \mathbf{p}^{(t+1)}, q^{(t+1)})$
using Metropolis-Hastings
until forever

$$\frac{\varepsilon}{4}|G(\boldsymbol{\theta}^{(t)})|^{-1}q^2 + q - q^{(t)} + \frac{\varepsilon}{2}\Bigg[-\mathscr{L}(\boldsymbol{\theta}^{(t)}) + gkT(1+\log s) + \frac{1+kT}{2}\log\left\{(2\pi)^D|\mathbf{G}(\boldsymbol{\theta})|\right\} - \frac{1}{2}\left[\frac{\mathbf{p}}{s^{(t)}}\right]^T G(\boldsymbol{\theta}^{(t)})^{-1}\left[\frac{\mathbf{p}}{s^{(t)}}\right]\Bigg] = 0$$

where we have omitted the $t + \varepsilon/2$ superscripts for clarity (i.e. $p_i = p_i^{(t+\varepsilon/2)}$).

4.2.2 The Stochastic Case

Typically when working with large datasets, computing the gradient of the log-likelihood over the entire dataset is very expensive. Therefore we resort to evaluating the gradients on a mini-batch of the data. The stochastic gradient of the log-likelihood can be written as

$$\nabla\tilde{\mathscr{L}}(\boldsymbol{\theta}) = \frac{|N|}{|\tilde{N}|}\sum_{x\in\tilde{N}}\nabla\log p(x|\boldsymbol{\theta}) + \nabla\log p(\boldsymbol{\theta}),$$

where N denotes the entire dataset, and $\tilde{N}$ denotes a random mini-batch.

From the dynamics in the deterministic case (4.5), we can see that there are two sources of stochastic noise in the minibatch setting from the two momenta terms: one in the equation

for $\dot{\mathbf{p}}$, where we have the extended position variable s multiplied by the gradient of the log-likelihood, and the second in the equation for $\dot{q}$, where we have the full log-likelihood term in H_{inner}. Note that the additive noise in the update for q is purely a function of $\boldsymbol{\theta}$, whereas the noise arising from the stochastic gradient in $\dot{\mathbf{p}}$ is multiplied by the extended position variable s. Therefore, following convention, if we write the stochastic terms (likelihood as well as gradient) as the corresponding full terms plus random noise, then we have the following expressions for the momenta dynamics in the stochastic setting:

$$\tilde{\dot{q}} = \dot{q} + N(0, 2A(\boldsymbol{\theta})), \quad \tilde{\dot{\mathbf{p}}} = \dot{\mathbf{p}} + N(0, 2\sqrt{s}B(\boldsymbol{\theta})).$$

This therefore turns the deterministic dynamics of (4.5) into a Langevin diffusion, with $A(\boldsymbol{\theta})$ and $\sqrt{s}B(\boldsymbol{\theta})$ acting as diffusion coefficients of standard Wiener processes.

As one might imagine, using these noisy terms in the dynamics without any correction leads to non-conservation of the total system energy; indeed, [CCG14] showed that under certain conditions the entropy of such a system would strictly increase with time. Therefore, one needs to introduce additional terms in the dynamics if the Hamiltonian (4.4) is to be conserved in the stochastic setting. To do so, we turn to the Fokker-Planck equation.

The Fokker-Planck corrections

The Fokker-Planck equation describes the evolution of the probability distribution of the parameters of a differential equation under stochastic forces. For a stochastic differential equation with diffusion coefficient $D(\boldsymbol{\theta})$, written as $\dot{\boldsymbol{\theta}} = f(\boldsymbol{\theta}) + N(0, 2D(\boldsymbol{\theta}))$, with the distribution of $\boldsymbol{\theta}$ being $p(\boldsymbol{\theta})$, the Fokker-Planck equation can be written as

$$\frac{\partial}{\partial t}\boldsymbol{p}(\boldsymbol{\theta}) = -\frac{\partial}{\partial \boldsymbol{\theta}}\left[f(\boldsymbol{\theta})p(\boldsymbol{\theta})\right] + \frac{\partial^2}{\partial \boldsymbol{\theta}^2}\left[D(\boldsymbol{\theta})p(\boldsymbol{\theta})\right], \tag{4.7}$$

where the notation $\frac{\partial^2}{\partial \boldsymbol{\theta}^2}$ denotes $\sum_{i,j} \frac{\partial}{\partial \theta_i}\frac{\partial}{\partial \theta_j}$.

Using this equation, we can derive the corrective terms for the stochastic noise in the dynamics (4.5). We propose the following corrected dynamics:

$$\begin{aligned}
\dot{\theta} &= \left(\frac{\mathbf{p}}{s}\right)^T \mathbf{G}(\boldsymbol{\theta})^{-1} \\
\dot{\mathbf{p}} &= s\frac{1}{2}\left(\frac{\mathbf{p}}{s}\right)^T \mathbf{G}(\boldsymbol{\theta})^{-1}\nabla\mathbf{G}(\boldsymbol{\theta})\mathbf{G}(\boldsymbol{\theta})^{-1}\left(\frac{\mathbf{p}}{s}\right) + s\frac{1}{2}q^2|\mathbf{G}(\boldsymbol{\theta})|^{-1}\mathrm{tr}\left(\mathbf{G}(\boldsymbol{\theta})^{-1}\nabla\mathbf{G}(\boldsymbol{\theta})\right) \\
&\quad - \sqrt{s}B(\boldsymbol{\theta})\left(\frac{\mathbf{p}}{s}\right)^T \mathbf{G}(\boldsymbol{\theta})^{-1} - \frac{s}{2}(1+kT)\mathrm{tr}\left(\mathbf{G}(\boldsymbol{\theta})^{-1}\nabla\mathbf{G}(\boldsymbol{\theta})\right) + s\nabla\tilde{\mathscr{L}}(\boldsymbol{\theta}) \\
\dot{s} &= sq|\mathbf{G}(\boldsymbol{\theta})|^{-1}, \quad \dot{q} = -gkT + \left(\frac{\mathbf{p}}{s}\right)^T \mathbf{G}(\boldsymbol{\theta})^{-1}\frac{\mathbf{p}}{s} - \tilde{H}_{\text{inner}} - A(\boldsymbol{\theta})sq\mathbf{G}(\boldsymbol{\theta})^{-1}.
\end{aligned} \tag{4.8}$$

Note that the correction terms consist of the Hamiltonian equations for the position variables with suitable multiplicative terms to cancel out the diffusion noise. This choice can be justified using the Fokker-Planck equation, as we prove below.

Theorem 2. *The dynamics defined in* (4.8) *leave the probability distribution defined by* $p(\boldsymbol{\theta},\boldsymbol{p},s,q) \propto \exp\left(-H(\boldsymbol{\theta},\boldsymbol{p},s,q)\right)$ *invariant.*

Proof. Let us start with the *deterministic* Hamiltonian dynamics, and replace the log-likelihood terms therein with their stochastic versions, without any corrections. Following the notation of [YA06], the dynamics can be represented in the following format:

$$\begin{bmatrix} \dot{\theta} \\ \dot{\mathbf{p}} \\ \dot{s} \\ \dot{q} \end{bmatrix} = -\begin{bmatrix} 0 & 0 & 0 & -I \\ 0 & 0 & I & 0 \\ 0 & -I & 0 & 0 \\ I & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \frac{\partial}{\partial s}H(\boldsymbol{\theta},\mathbf{p},s,q) \\ \frac{\partial}{\partial q}H(\boldsymbol{\theta},\mathbf{p},s,q) \\ \frac{\partial}{\partial \boldsymbol{\theta}}H(\boldsymbol{\theta},\mathbf{p},s,q) \\ \frac{\partial}{\partial \mathbf{p}}H(\boldsymbol{\theta},\mathbf{p},s,q) \end{bmatrix} + \boldsymbol{N} \tag{4.9}$$

where $\boldsymbol{N} = [0, N(0, 2\sqrt{s}B(\boldsymbol{\theta})), 0, N(0, 2A(\boldsymbol{\theta}))]^T$. Let us denote the anti-symmetric matrix above by X. Then, denoting $\nabla = [\partial/\partial\boldsymbol{\theta}; \partial/\partial\mathbf{p}; \partial/\partial s; \partial/\partial q]$, it is easy to see that $\mathrm{tr}\left\{\nabla^T\nabla X y\right\} = 0$ for any $y(\boldsymbol{\theta},\mathbf{p},s,q)$.

Therefore the right hand side of the Fokker-Planck equation (4.7) can be written as

$$\begin{aligned}
&\mathrm{tr}\nabla^T\left\{p(\boldsymbol{\theta},\mathbf{p},s,q)X\nabla H\right\} + \mathrm{tr}\left\{\nabla^T D\nabla p(\boldsymbol{\theta},\mathbf{p},s,q)\right\} \\
=&\mathrm{tr}\nabla^T\left\{p(\boldsymbol{\theta},\mathbf{p},s,q)X\nabla H\right\} + \mathrm{tr}\left\{(D+X)\nabla^T\nabla p(\boldsymbol{\theta},\mathbf{p},s,q)\right\}.
\end{aligned}$$

Here we have used the shorthand ∇H to refer to the second matrix on the right hand side of equation (4.9), and D contains the diffusion terms from the stochastic noise (see §C of the supplementary for the exact formulation).

Note that $\nabla p = -p\nabla H$, since $p \propto \exp(-H)$. Therefore, if we simply replace X with $D+X$ in (4.9), the right hand side of the Fokker-Plank equation reduces to zero. Using $D+X$ in (4.9) is equivalent to the dynamics (4.8). □

As mentioned before, we use the generalized leapfrog algorithm to discretize the continuous differential equations of motion. The generalized leapfrog algorithm is a composition of a symplectic first-order Euler integrator with its adjoint. We describe the discretized version of the dynamics (4.8) in §B of the supplementary.

4.3 Experiments

4.3.1 Estimation of 1D Gaussian Distribution

We start off with a synthetic experiment on learning the parameters of a one-dimensional Gaussian distribution. We generate 5000 points from a standard normal distribution, and attempt to learn the mean and the variance using the discretized stochastic algorithm based on the dynamics (4.8). We call this algorithm stochastic gradient Riemann Nosé-Poincaré Hamiltonian Monte Carlo (SGR-NPHMC). We compare it to the SG-NHT algorithm of [DFB$^+$14]. Extensive comparisons of SG-NHT with related techniques like stochastic gradient Hamiltonian Monte Carlo and Langevin Dynamics have already been performed in the literature, hence we do not conduct comparisons with those methods here.

For both SGR-NPHMC and SG-NHT, we use normal-Wishart priors on the mean and precision; the posterior distribution is proportional to $p(\mu,\tau|\mathbf{X}) \propto N(\mathbf{X}|\mu,\tau)\mathscr{W}(\tau|1,1)$, where τ denotes the precision, and $\mathscr{W}$ denotes the Wishart distribution. We run both algorithms for 10^5 iterations and discard the first 5000 "burn-in" iterations. For our Riemannian

algorithm we use the observed Fisher information plus the negative Hessian of the prior as the metric tensor, and perform one fixed point iteration to solve the implicit system of equations. For both algorithms we use 10 leapfrog iterations. Learning rates are fixed to 1e-3 and batchsizes to 100 for both algorithms.

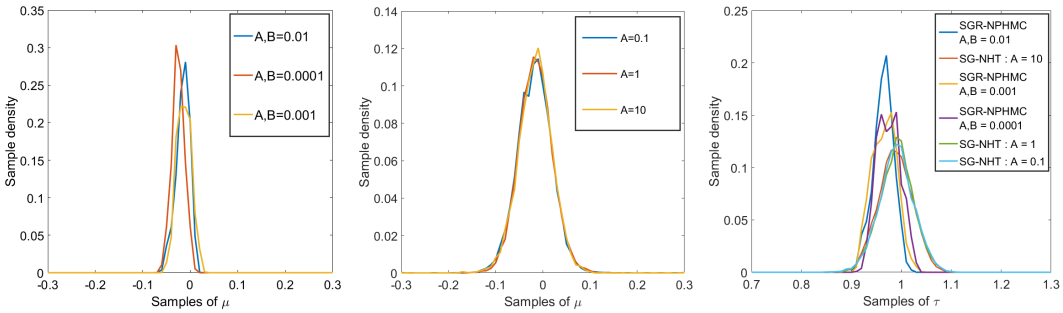
In Figure 4.1 we demonstrate the sensitivity of the algorithms to different values of the stochastic noise correction terms. The post-burnin samples of μ generated by both algorithms for various values of these terms are plotted in Figures 4.1a and 4.1b. 4.1c. Figure 4.1c shows the corresponding precision samples. We can see that SGR-NPHMC is as robust to stochastic noise as SG-NHT, and both algorithms generate aceptable samples of μ and τ post-burnin.

However the sampling trajectories in Figures 4.2a and 4.2b tell a different story. We see that SG-NHT overshoots the target value of μ by a large margin, as well as having a higher spread of the post-burnin samples. In contrast, SGR-NPHMC follows a more direct path to the target, and generates a tighter set of samples. This behavior can be attributed to the Riemann geometry cues and the resulting implicit system of update equations.

The higher variance in the samples shows up in the RMSE for the parameters. We show these numbers along with the autocorrelation times for both algorithms in Tables 4.1 and 4.2. As seen in the qualitative sample trajectories, the SGR-NPHMC generates samples with lower RMSE for all values of the noise corrector terms.

4.3.2 Parameter Estimation in Bayesian Logistic Regression

Our next experiment is on learning the parameters in a synthetic two-dimensional Bayesian logistic regression task. In this experiment we first generate 5000 datapoints from two bivariate normal distributions with means at $[1,-1]$ and $[-1,1]$ and unit covariances, and then use a linear classifier with weights $(w_1,w_2) = [1,-1]$ to bag the points into two



(a) (b) (c)

Figure 4.1: Density plots for the mean (μ) and precision (τ) samples obtained from SGR-NPHMC and SG-NHT runs on the synthetic Gaussian dataset. Plots (a) and (b) show the sample densities of μ for SGR-NPHMC and SG-NHT respectively. Plot (c) shows the sample densities of τ for both algorithms. The true values were $\mu = 0, \tau = 1$. See the text for experimental details.

Table 4.1: RMSE and auto-correlation times of the sampled means, precisions from SGR-NPHMC runs on synthetic Gaussian data.

{A,B}	RMSE (μ)	RMSE (τ)	A.T. (μ)
0.01	0.0240	0.0328	14.8999
0.001	0.0244	0.0466	13.6332
0.0001	0.0289	0.0433	2.5899

classes. We then estimate the classifier weights using Bayesian logistic regression. As with the previous experiment, we compare SGR-NPHMC and SG-NHT in terms of accuracy and autocorrelation time. We run both algorithms for 10^5 iterations, discard the first 5000 samples and use the rest to compute these metrics. SGR-NPHMC achieves lower RMSE for the parameters in this case as well, as seen in the corresponding tables provided in §D.1 of the supplementary. The sample trajectories shown in Figures 4.3a and 4.3b paint a similar picture to that of the previous section; SG-NHT overshoots by a wide margin before

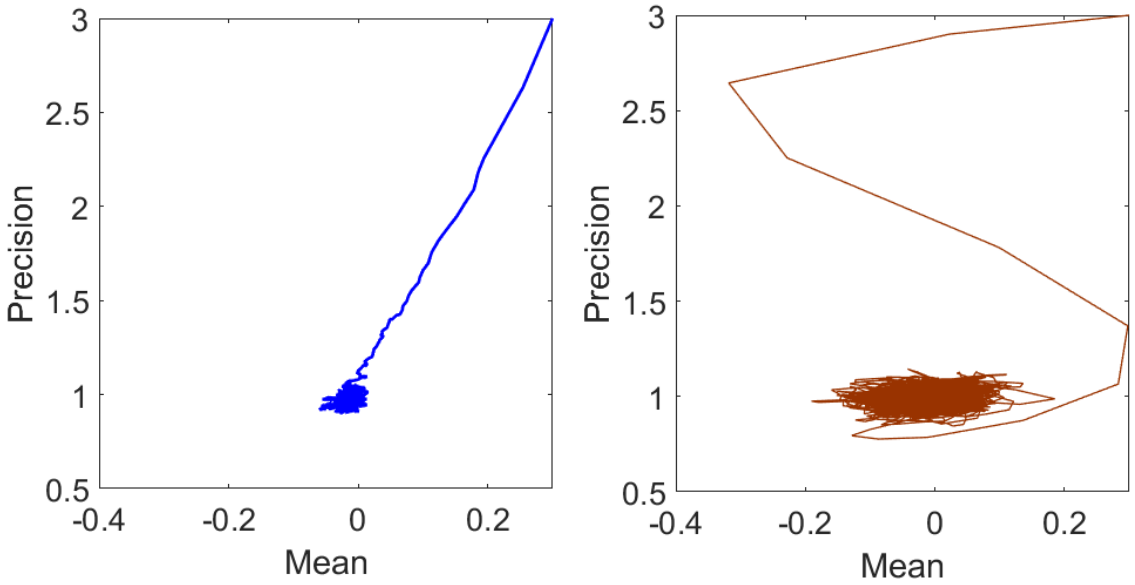


Figure 4.2: Samples trajectories for μ and τ from (a) SGR-NPHMC and (b) SG-NHT runs on the synthetic Gaussian dataset. Both algorithms were initialized with $\mu_0 = 0.3, \tau_0 = 3$. For the former we used $\{A, B\} = 0.001$, and for the latter we had $A = 1$. The true values were $\mu = 0, \tau = 1$. Note the convergence patterns and sample spread of the two algorithms.

converging, and has higher sample variance as well. SGR-NPHMC follows a more efficient path to convergence, and post-convergence sample variance is lower as well.

4.3.3 Topic Modeling using Hierarchical Gamma Processes

For this experiment we compare the algorithms in a high-dimensional topic modeling scenario using hierarchical Gamma processes. In particular, we use the Poisson factor analysis framework of [ZC15]. We model the observed counts of V vocabulary terms in N documents as $\mathbf{D}_{V\times N} = \text{Poi}(\mathbf{\Phi\Theta})$, where $\mathbf{\Phi}_{V\times K}$ is the factor load matrix that encodes the relative importance of the vocabulary terms in the K latent topics, and $\mathbf{\Theta}_{K\times N}$ models the counts of the topics in the documents.

Table 4.2: RMSE and auto-correlation times of the sampled means and precisions from SG-NHT runs on synthetic Gaussian data.

{A}	RMSE (μ)	RMSE (τ)	A.T. (μ)
0.1	0.0364	0.0386	13.5715
1	0.0375	0.0471	17.2241
10	0.0365	0.0416	13.5715

Table 4.3: Test perplexities on 20-Newsgroups and Reuters datasets.

METHOD	MODEL	20-NEWSGROUPS	REUTERS
GIBBS	γNB	763	-
GIBBS	NB-LDA	788	-
SG-NHT	γGP	758	929
SGR-NPHMC	γNB	752	930
SGR-NPHMC	γGP	723	904

We put a Dirichlet prior on the columns of $\mathbf{\Phi}$ using normalized Gamma variables: $\phi_{v,k} = \frac{\gamma_v}{\sum_v \gamma_v}$, with $\gamma_v \sim \Gamma(\alpha, 1)$. Then we have $\theta_{n,k} \sim \Gamma(r_k, \frac{p_j}{1-p_j})$, where the document-specific mixing probabilities p_j have $\beta(a_0, b_0)$ priors. Next, we use two different formulations of r_k: (a) we set r_ks to the weights of a discrete Gamma process with equal atom weights, as in [ZC15]; and (b) we set r_ks to the atom weights generated by the constructive Gamma process definition of [RK15]. See the respective papers for the details of the constructions. We call these two formulations γNB, and γGP respectively.

We use two public datasets for this experiment, the 20-Newsgroups and Reuters Corpus Volume 1 corpora from [SSH13]. The first has a vocabulary of 2,000 words spread over 18,845 documents. The second has 804,414 documents and a vocabulary of size 10,000. We use the same training/validation/test split as [GCH$^+$15], where the 20 Newsgroups dataset is

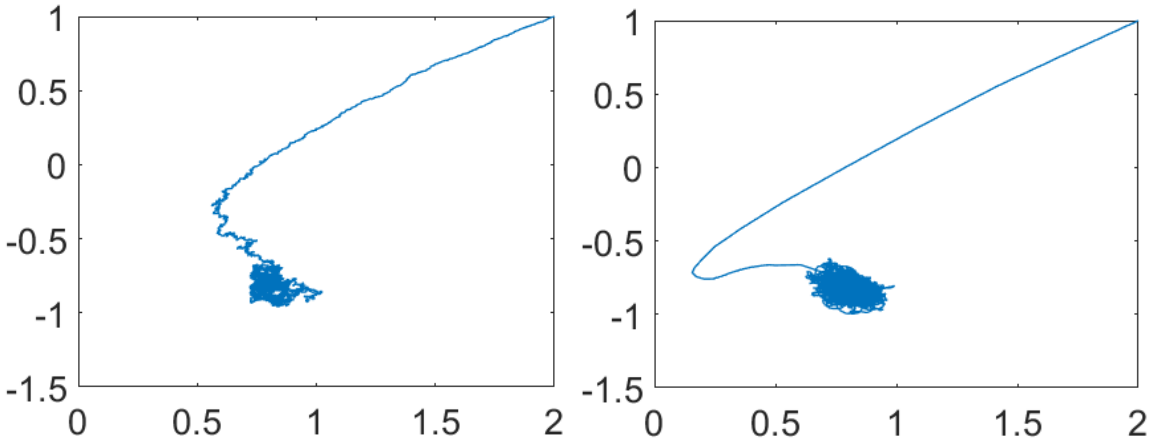


Figure 4.3: Samples trajectories for w_1 and w_2 from (a) SGR-NPHMC and (b) SG-NHT runs on the synthetic Bayesian logistic regression dataset. Both algorithms were initialized with $w_1 = 2, w_2 = 1$. For the former we used $\{A, B\} = 0.001$, and for the latter we had $A = 1$. The true values were $w_1 = 1, w_2 = -1$. Note the convergence patterns of the two algorithms, and the spread of the samples thereafter.

split chronologically into 11,314 training and 7,531 test documents, and the Reuters dataset into 794,414 training and 10,000 test documents. After training the stochastic algorithms, following standard methodology we learn document-specific parameters from 80% of the words in the test set, and calculate test perplexities on the remaining 20%. The perplexity formulation is detailed in §D.2 of the supplementary.

For SGR-NPHMC and SG-NHT on the γGP model, we run three parallel NPHMC chains; one each for the two constituent parameters of the atom weights (E_ks and T_ks) and one for the hyperparameters (α, γ and c). See §4.1 of [RK15] for the exact formulation. We estimate the ϕs using Riemann Hamiltonian Monte Carlo updates [GC11], as we found it to mix better than Gibbs sampling for the stochastic algorithms. For all Riemannian HMC chains we use the diagonal metric tensor $G(\boldsymbol{\theta}) = \text{diag}(\boldsymbol{\theta})^{-1}$, as first studied in [PT13] (see §4.2.1.2 for the resulting dynamics). We used $K = 200$ latent topics for all algorithms. For SGR-NPHMC we set the learning rates of all three NPHMC chains to $1e$-4, and for SG-NHT

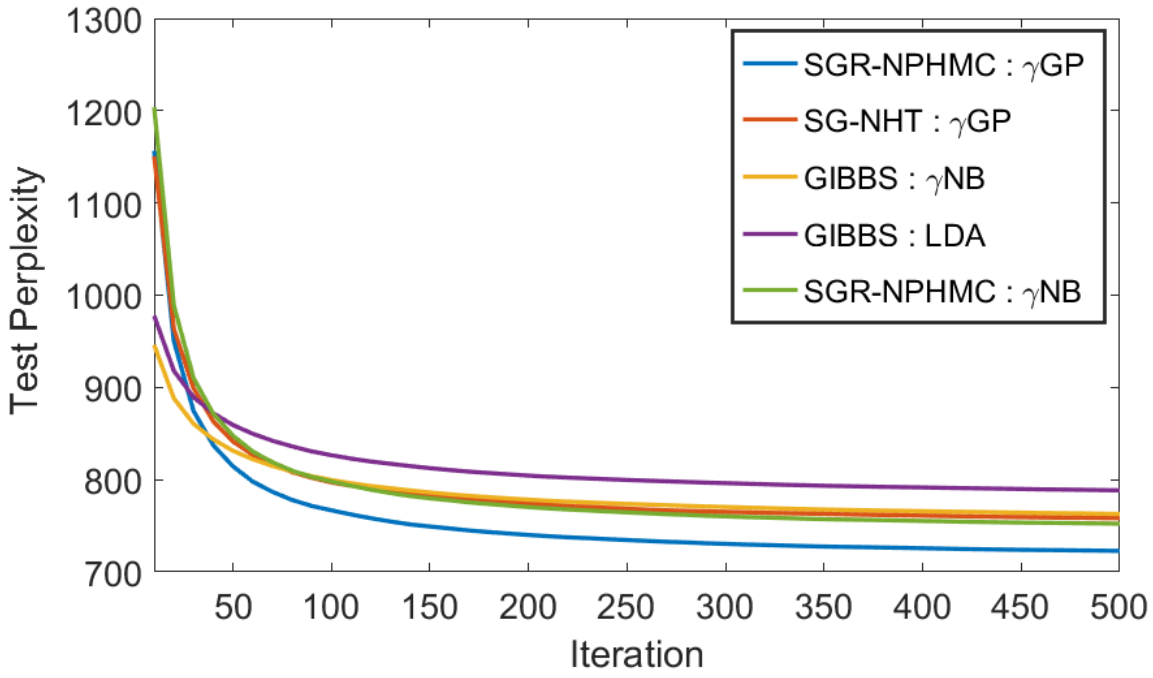


Figure 4.4: Test perplexities as a function of post-burnin iterations for the 20-Newsgroups dataset.

we use a stable learning rate of $1e$-6. Batchsize was set to 100 for both algorithms. We used 2,000 iterations for burn-in and collected samples for test perplexity evaluation thereafter.

Figure 6.1 shows the perplexities evaluated on the 20-Newsgroups dataset. The perplexities at the end of the test runs for both 20-Newsgroups and Reuters are shown in Table **??**. We can see the SGR-NPHMC algorithms for both γNB and γGP models outperforming SG-NHT for γGP on 20-newsgroups. For the larger Reuters dataset, the γGP-based SGR-NPHMC performs best, followed by SG-NHT and the γNB-based SGR-NPHMC.

4.4 Conclusion

We have proposed a novel Hamiltonian MCMC algorithm using a modified Nosé-Poincaré Hamiltonian augmented with Riemann preconditioning for both real and extended

momenta, as well as correction terms arising from the Fokker-Planck equations to ensure sampling from the canonical ensemble in the presence of stochastic noise. We have derived a discretized sampler using the generalized leapfrog algorithm, and have shown robust performance in synthetic and high dimensional real-world datasets.

4.5 Proof Details and Experimental Addenda

4.5.1 Proof of Theorem 1

The Riemann-augmented Nosé-Poincaré Hamiltonian can be written as

$$H(\boldsymbol{\theta},\mathbf{p},s,q) = s\Bigg(-\mathscr{L}(\boldsymbol{\theta}) + \frac{1}{2}\left(\frac{\mathbf{p}}{s}\right)^T \mathbf{G}(\boldsymbol{\theta})^{-1}\frac{\mathbf{p}}{s} + \frac{1}{2}|\mathbf{G}(\boldsymbol{\theta})|^{-1}q^2 + \frac{1+kT}{2}\log\left\{(2\pi)^D|\mathbf{G}(\boldsymbol{\theta})|\right\} + gkT\log s - H_0\Bigg). \tag{4.10}$$

We need to prove the following about this Hamiltonian:

Theorem 1. *The Riemannian Nosé-Poincaré Hamiltonian defined in* (4.10) *generates samples from the canonical ensemble.*

Proof. First we denote $H_{gc}(\boldsymbol{\theta},\mathbf{p}) = -\mathscr{L}(\boldsymbol{\theta}) + \frac{1}{2}\mathbf{p}^T\mathbf{G}(\boldsymbol{\theta})^{-1}\mathbf{p}$.

We will show that we can integrate out s,q from $p(\boldsymbol{\theta},\mathbf{p},s,q) \propto \exp\left(-H(\boldsymbol{\theta},\mathbf{p},s,q)\right)$ to get $p(\boldsymbol{\theta},\mathbf{p}) \propto \exp\left(-H_{gc}(\boldsymbol{\theta},\mathbf{p})/kT\right)$. The integration of s essentially follows [BLL99].

The probability of any $(\boldsymbol{\theta},\mathbf{p},s,q)$ can be written as

$$\begin{aligned}
&p(\boldsymbol{\theta},\mathbf{p},s,q) \propto \delta[H - H_0] \\
&\propto \delta\left[s\Big(H_{gc}\left(\boldsymbol{\theta},\left(\frac{\mathbf{p}}{s}\right)\right) + \frac{1}{2}|\mathbf{G}(\boldsymbol{\theta})|^{-1}q^2 + \frac{1+kT}{2}\log\left\{(2\pi)^D|\mathbf{G}(\boldsymbol{\theta})|\right\} + gkT\log s - H_0\Big)\right] \\
&\propto s^{N_f}\delta\left[s\Big(H_{gc} + \frac{1}{2}|\mathbf{G}(\boldsymbol{\theta})|^{-1}q^2 + \frac{1+kT}{2}\log\left\{(2\pi)^D|\mathbf{G}(\boldsymbol{\theta})|\right\} + gkT\log s - H_0\Big)\right].
\end{aligned}$$

See [LR05] for the details of the last step.

First we integrate out s. The argument of the δ-function has a root at $s_0 = \exp\Bigg[-\frac{1}{gkT}\left(H_{gc} + \frac{1}{2}|\mathbf{G}(\boldsymbol{\theta})|^{-1}q^2 + \frac{kT}{2}\log\left\{(2\pi)^D|\mathbf{G}(\boldsymbol{\theta})|\right\} - H_0\right)\Bigg]$. Therefore, we have $p(\boldsymbol{\theta},\mathbf{p},s,q)$

$$\begin{aligned}
&\propto \frac{s^{N_f}}{gkT}\delta\left[s - \exp\left(-\frac{1}{gkT}\left(H_{gc} + \frac{1}{2}|\mathbf{G}(\boldsymbol{\theta})|^{-1}q^2 + \frac{kT}{2}\log\left\{(2\pi)^D|\mathbf{G}(\boldsymbol{\theta})|\right\} - H_0\right)\right)\right] \\
&\propto \exp\left(\frac{N_f H_0}{gkT}\right)\exp\left(-\frac{N_f}{gkT}\left(H_{gc} + \frac{1}{2}|\mathbf{G}(\boldsymbol{\theta})|^{-1}q^2 + \frac{kT}{2}\log\left\{(2\pi)^D|\mathbf{G}(\boldsymbol{\theta})|\right\}\right)\right) \\
&\propto \exp\left(-\frac{1}{kT}\left(H_{gc} + \frac{1}{2}|\mathbf{G}(\boldsymbol{\theta})|^{-1}q^2 + \frac{kT}{2}\log\left\{(2\pi)^D|\mathbf{G}(\boldsymbol{\theta})|\right\}\right)\right), \text{ using } g = N_f.
\end{aligned}$$

See [LR05] for the details of the first step. With s integrated out, we are left with

$$p(\boldsymbol{\theta}, \mathbf{p}, q) \propto \exp\left(-H_{gc}/kT\right) \exp\left[-\frac{1}{2kT}|\mathbf{G}(\boldsymbol{\theta})|^{-1}q^2 - \frac{1}{2}\log\left\{(2\pi)^D|\mathbf{G}(\boldsymbol{\theta})|\right\}\right].$$

We can easily integrate out q from the second exponential term on the right to get the desired form for $p(\boldsymbol{\theta}, \mathbf{p})$. □

4.5.2 Discretized Dynamics

4.5.2.1 Generalized Leapfrog in the Stochastic case

We propose the following dynamics to incorporate stochastic noise correction terms into the deterministic updates:

$$\begin{aligned}
\dot{\theta} &= \left(\frac{\mathbf{p}}{s}\right)^T \mathbf{G}(\boldsymbol{\theta})^{-1} \\
\dot{\mathbf{p}} &= s\frac{1}{2}\left(\frac{\mathbf{p}}{s}\right)^T \mathbf{G}(\boldsymbol{\theta})^{-1}\nabla\mathbf{G}(\boldsymbol{\theta})\mathbf{G}(\boldsymbol{\theta})^{-1}\left(\frac{\mathbf{p}}{s}\right) + s\nabla\tilde{\mathscr{L}}(\boldsymbol{\theta}) \\
&\quad + s\frac{1}{2}q^2\mathbf{G}(\boldsymbol{\theta})^{-1}\nabla\mathbf{G}(\boldsymbol{\theta})\mathbf{G}(\boldsymbol{\theta})^{-1} - \sqrt{s}B(\boldsymbol{\theta})\left(\frac{\mathbf{p}}{s}\right)^T \mathbf{G}(\boldsymbol{\theta})^{-1} \\
&\quad - \frac{s}{2}(1+kT)\mathrm{tr}\left(\mathbf{G}(\boldsymbol{\theta})^{-1}\nabla\mathbf{G}(\boldsymbol{\theta})\right) \\
\dot{s} &= sq\mathbf{G}(\boldsymbol{\theta})^{-1} \\
\dot{q} &= -gkT + \left(\frac{\mathbf{p}}{s}\right)^T \mathbf{G}(\boldsymbol{\theta})^{-1}\frac{\mathbf{p}}{s} - \tilde{H}_{\text{inner}} - A(\boldsymbol{\theta})sq\mathbf{G}(\boldsymbol{\theta})^{-1}.
\end{aligned} \tag{4.11}$$

As mentioned in the main chapter, we use the generalized leapfrog algorithm to discretize the continuous differential equations. The generalized leapfrog algorithm is a composition of a symplectic first-order Euler integrator with its adjoint. For the dynamics (4.11), the

update equations can be written as

$$
\begin{aligned}
p_i^{(t+\varepsilon/2)} &= p_i^{(t)} - \frac{\varepsilon}{2}\nabla_{\theta_i} H\left(\boldsymbol{\theta}^{(t)}, s^{(t)}, \mathbf{p}^{(t+\varepsilon/2)}, q^{(t+\varepsilon/2)}\right) \\
q^{(t+\varepsilon/2)} &= q^{(t)} - \frac{\varepsilon}{2}\Bigg[As^{(t)}q^{(t)}G(\boldsymbol{\theta}^{(t)})^{-1} + \tilde{H}_{\text{inner}} + gkT \\
&\qquad - \left(\frac{\mathbf{p}^{(t+\varepsilon/2)}}{s^{(t)}}\right)^T G(\boldsymbol{\theta}^{(t)})^{-1}\left(\frac{\mathbf{p}^{(t+\varepsilon/2)}}{s^{(t)}}\right)\Bigg] \\
\theta_i^{(t+\varepsilon} &= \theta_i^{(t)} + \frac{\varepsilon}{2}\left[\left(\frac{\mathbf{p}^{(t+\varepsilon/2)}}{s^{(t)}}\right)^T G(\boldsymbol{\theta}^{(t)})^{-1} + \left(\frac{\mathbf{p}^{(t+\varepsilon/2)}}{s^{(t+\varepsilon)}}\right)^T G(\boldsymbol{\theta}^{(t+\varepsilon)})^{-1}\right]_i \\
s^{(t+\varepsilon)} &= s^{(t)} + \frac{\varepsilon}{2}\left[s^{(t)}q^{(t+\varepsilon/2)}|G(\boldsymbol{\theta}^{(t)})|^{-1} + s^{(t+\varepsilon)}q^{(t+\varepsilon/2)}|G(\boldsymbol{\theta}^{(t+\varepsilon)})|^{-1}\right] \\
p_i^{(t+\varepsilon)} &= p_i^{(t+\varepsilon/2)} - \frac{\varepsilon}{2}\nabla_{\theta} H\left(\boldsymbol{\theta}^{(t+\varepsilon)}, s^{(t+\varepsilon)}, \mathbf{p}^{(t+\varepsilon/2)}, q^{(t+\varepsilon/2)}\right) \\
q^{(t+\varepsilon)} &= q^{(t+\varepsilon/2)} - \frac{\varepsilon}{2}\Bigg[As^{(t+\varepsilon)}q^{(t+\varepsilon/2)}G(\boldsymbol{\theta}^{(t+\varepsilon)})^{-1} + \tilde{H}_{\text{inner}} + gkT \\
&\qquad - \left(\frac{\mathbf{p}^{(t+\varepsilon/2)}}{s^{(t+\varepsilon)}}\right)^T G(\boldsymbol{\theta}^{(t+\varepsilon)})^{-1}\left(\frac{\mathbf{p}^{(t+\varepsilon/2)}}{s^{(t+\varepsilon)}}\right)\Bigg]
\end{aligned}
\tag{4.12}
$$

where $\nabla_{\theta_i} H(\boldsymbol{\theta}, s, \mathbf{p}, q)$

$$
\begin{aligned}
&= \frac{\varepsilon}{2}\Bigg[-\frac{1}{2}s\left(\frac{\mathbf{p}}{s}\right)^T G(\boldsymbol{\theta})^{-1}\left(\frac{\partial}{\partial\theta_i}G(\boldsymbol{\theta})\right)G(\boldsymbol{\theta})^{-1}\left(\frac{\mathbf{p}}{s}\right) - s\frac{q^2}{2}|G(\boldsymbol{\theta})|^{-1}\text{tr}\left\{G(\boldsymbol{\theta})^{-1}\frac{\partial}{\partial\theta_i}G(\boldsymbol{\theta})\right\} \\
&\quad + \sqrt{s}B\left(\frac{\mathbf{p}}{s}\right)^T G(\boldsymbol{\theta})^{-1} + \frac{s}{2}(1+kT)\text{tr}\left\{G(\boldsymbol{\theta})^{-1}\frac{\partial}{\partial\theta_i}G(\boldsymbol{\theta})\right\} - s\frac{\partial}{\partial\theta_i}\tilde{\mathscr{L}}(\boldsymbol{\theta})\Bigg].
\end{aligned}
$$

4.5.3 Proof of Theorem 2

The Fokker-Planck equation describes the evolution of the probability distribution of the parameters of a differential equation under stochastic forces. For a stochastic differential equation with diffusion coefficient $D(\boldsymbol{\theta})$, written as $\dot{\boldsymbol{\theta}} = f(\boldsymbol{\theta}) + N(0, 2D(\boldsymbol{\theta}))$, with the distribution of $\boldsymbol{\theta}$ being $p(\boldsymbol{\theta})$, the Fokker-Planck equation can be written as

$$
\frac{\partial}{\partial t}\boldsymbol{p}(\boldsymbol{\theta}) = -\frac{\partial}{\partial\boldsymbol{\theta}}\left[f(\boldsymbol{\theta})p(\boldsymbol{\theta})\right] + \frac{\partial^2}{\partial\boldsymbol{\theta}^2}\left[D(\boldsymbol{\theta})p(\boldsymbol{\theta})\right], \tag{4.13}
$$

where the notation $\frac{\partial^2}{\partial\boldsymbol{\theta}^2}$ denotes $\sum_{i,j}\frac{\partial}{\partial\theta_i}\frac{\partial}{\partial\theta_j}$.

Theorem 2. *The dynamics defined in* (4.11) *leave the probability distribution defined by* $p(\boldsymbol{\theta},\boldsymbol{p},s,q) \propto \exp\left(-H(\boldsymbol{\theta},\boldsymbol{p},s,q)\right)$ *invariant.*

Proof. Let us start with the *deterministic* Hamiltonian dynamics, and replace the log-likelihood terms therein with their stochastic versions, without any corrections. Following the notation of [YA06], the dynamics can be represented in the following format:

$$\begin{bmatrix} \dot{\boldsymbol{\theta}} \\ \dot{\mathbf{p}} \\ \dot{s} \\ \dot{q} \end{bmatrix} = - \begin{bmatrix} 0 & 0 & 0 & -I \\ 0 & 0 & I & 0 \\ 0 & -I & 0 & 0 \\ I & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \frac{\partial}{\partial s} H(\boldsymbol{\theta}, \mathbf{p}, s, q) \\ \frac{\partial}{\partial q} H(\boldsymbol{\theta}, \mathbf{p}, s, q) \\ \frac{\partial}{\partial \boldsymbol{\theta}} H(\boldsymbol{\theta}, \mathbf{p}, s, q) \\ \frac{\partial}{\partial \mathbf{p}} H(\boldsymbol{\theta}, \mathbf{p}, s, q) \end{bmatrix} + \boldsymbol{N} \tag{4.14}$$

where $\boldsymbol{N} = [0, N(0, 2\sqrt{s}B(\boldsymbol{\theta})), 0, N(0, 2A(\boldsymbol{\theta}))]^T$. Let us denote the anti-symmetric matrix above by X. Then, denoting $\nabla = [\partial/\partial\boldsymbol{\theta}; \partial/\partial\mathbf{p}; \partial/\partial s; \partial/\partial q]$, it is easy to see that $\mathrm{tr}\left\{\nabla^T \nabla X y\right\} = 0$ for any $y(\boldsymbol{\theta}, \mathbf{p}, s, q)$.

Therefore the right hand side of the Fokker-Planck equation (4.13) can be written as

$$\begin{aligned} &- \mathrm{tr}\nabla^T \left\{p(\boldsymbol{\theta}, \mathbf{p}, s, q) X \nabla H\right\} + \mathrm{tr}\left\{\nabla^T D \nabla p(\boldsymbol{\theta}, \mathbf{p}, s, q)\right\} \\ =& - \mathrm{tr}\nabla^T \left\{p(\boldsymbol{\theta}, \mathbf{p}, s, q) X \nabla H\right\} + \mathrm{tr}\left\{(D+X)\nabla^T \nabla p(\boldsymbol{\theta}, \mathbf{p}, s, q)\right\}. \end{aligned}$$

Here we have used the shorthand ∇H to refer to the second matrix on the right hand side of equation (4.14), and

$$D = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \sqrt{s}B(\boldsymbol{\theta}) \\ 0 & 0 & 0 & 0 \\ 0 & A(\boldsymbol{\theta}) & 0 & 0 \end{bmatrix}.$$

Note that $\nabla p = -p\nabla H$, since $p \propto \exp(-H)$. Therefore, if we simply replace X with $D+X$ in (4.14), the right hand side of the Fokker-Plank equation reduces to zero. Using $D+X$ in (4.14) is equivalent to the dynamics (4.11). □

As an aside, one can intuitively see that it is possible to incorporate any arbitrary dynamics into this formulation, by making appropriate assumptions about the diffusion terms in the noise and making corresponding corrections to the deterministic dynamics. Indeed, this fact has been investigated more thoroughly in [MCF15].

4.5.4 Additional Experimental Details

4.5.4.1 Parameter Estimation in Bayesian Logistic Regression

Tables 4.4 and 4.5 show the RMSE for the two parameters w_1 and w_2 for the synthetic Bayesian logistic regression experiment, for SGR-NPHMC and SG-NHT respectively.

Table 4.4: RMSE and auto-correlation times of the parameter samples from SGR-NPHMC runs on the synthetic Bayesian logistic regression dataset.

{A,B}	RMSE (w_1)	RMSE (w_2)
0.01	0.2684	0.1833
0.001	0.1927	0.1932
0.0001	0.1965	0.2125

Table 4.5: RMSE and auto-correlation times of the parameter samples from SG-NHT runs on the synthetic Bayesian logistic regression dataset.

{A}	RMSE (w_1)	RMSE (w_2)
0.1	0.2071	0.1884
1	0.2023	0.1850
10	0.2151	0.1907

4.5.4.2 Perplexity for Topic Modeling

We use the perplexity metric from [ZC15], where it is defined as

$$\text{Perplexity} = \exp\left(-\frac{1}{y_{\cdot\cdot}}\sum_{n=1}^{N_{test}}\sum_{v=1}^{V} y_{nv}\log m_{nv}\right)$$

where y_{nv} =the count of vocabulary term v in held-out test document n, and $y_{\cdot\cdot} = \sum_{n=1}^{N_{test}}\sum_{v=1}^{V} y_{nv}$.

m_{nv} is the mean of the collected samples of $\boldsymbol{\Theta} \times \boldsymbol{\Phi}$ normalized over the vocabulary, defined as

$$m_{nv} = \sum_{s=1}^{S} \sum_{k=1}^{K} \phi_{vk}^{(s)} \theta_{kn}^{(s)} \Big/ \sum_{v=1}^{V} \sum_{s=1}^{S} \sum_{k=1}^{K} \phi_{vk}^{(s)} \theta_{kn}^{(s)}.$$

Here s and k index the samples and latent topics respectively.

Chapter 5: Stochastic Quasi-Newton Optimization on Riemannian Manifolds

Optimization algorithms are a mainstay in machine learning research, underpinning solvers for a wide swath of problems ranging from linear regression and SVMs to deep learning. Consequently, scaling such algorithms to large scale datasets while preserving theoretical guarantees is of paramount importance. An important challenge in this field is designing scalable algorithms for optimization problems in the presence of constraints on the search space, a situation all too often encountered in real life. One approach to handling such constrained optimization problems on vector spaces is to reformulate them as optimization tasks on a suitable Riemannian manifold, with the constraints acting as manifold parametrization. Often, the problems can be shown to possess desirable geometric properties like convexity with respect to distance-minimizing geodesics on the manifold, leading to provably efficient optimization algorithms [AMS08, ZS16, SH15, RW12]. These ideas can then be combined with stochastic optimization techniques influenced by [RM51], to deal with large datasets with theoretical convergence guarantees. See [Bon13, ZRS16] for recent examples. For instance, we can consider the problem of computing leading eigenvectors in the PCA setting [Sha15] with unit-norm constraints. Projection-based strategies are normally used for this kind of problems [Oja92], but alternating between solving and projecting can be prohibitively expensive in high dimensions. However, the unit-norm constraint can be used to cast the eigenvector problem into an unconstrained

optimization scenario on the unit sphere, which happens to be one of the most well-behaved Riemannian manifolds.

Once the problems have been cast onto manifolds, one would want fast optimization algorithms that potentially use stochastic minibatches to deal with very large datasets. Such algorithms operating in Euclidean space have been widely researched in the optimization literature, but their development for Riemannian manifolds has been limited so far. In particular, one should note that the convergence speed limitations of unconstrained stochastic algorithms in the Euclidean case apply to manifold optimization as well; for instance a straightforward port of stochastic gradient descent to Riemannian manifolds [ZS16] attains the same sublinear convergence seen in Euclidean space. There has been extensive work in the Euclidean domain using variance-reduced gradients to address this issue, with the aim of improving convergence rates by explicitly reducing the variance of stochastic gradients with suitably spaced full-gradient evaluations [Sha15, JZ13]. Another nice advantage of this technique is the removal of the need for decaying learning rates for proving convergence, thereby solving the sublinearity issue as well sidestepping the nontrivial task of selecting an appropriate decay rate for SGD-like algorithms in large-scale optimization scenarios. Researchers have begun porting these methods to the manifold optimization domain, with a stochastic first-order variance reduced technique [ZRS16] showing robust convergence guarantees for both convex and nonconvex problems on geodesically complete manifolds.

Another complementary approach to improving convergence rates is of course using second-order updates for the iterates. In the Euclidean setting, one can show quadratic convergence rates for convex problems using Newton iterations, but these tend to be prohibitively expensive in high-dimensional big-data settings due to the need to store and invert the Hessian matrix. This limitation has led to the development of quasi-Newton methods, most notably L-BFGS [LN89], which uses lower-order terms to approximate the inverse

Hessian. The curvature information provided by the Hessian estimate allows superlinear convergence in ideal settings [NW06]. While widely used for small-to-medium scale problems, adoption of these methods for big data problems has been limited, since the second order updates can be prohibitively expensive to compute in these situations. However, most optimization algorithms in the literature that use stochastic minibatching techniques to deal with large datasets are modifications of first order gradient-descent [BL04, Bot10] with relatively slower convergence in practical situations. This has recently begun to be addressed, with researchers devising stochastic variants of the L-BFGS technique [BHNS14], with straightforward convergence analyses. This has also been combined with variance reduction techniques and shown to have a linear convergence rate for convex problems in Euclidean space [MNJ16]. Our work in this paper is in a similar vein: we study quasi-Newton L-BFGS updates with stochastic variance reduction techniques for optimization problems on Riemannian manifolds, and analyze their convergence behavior for convex and nonconvex functions.

On the topic of connecting the algorithm in this chapter to the thesis statement, one can consider applying it to a variety of constrained optimization problems that appear in Bayesian methods, both parametric and nonparametric. As an example of a parametric application, it is known that an asymptotic analysis of probabilistic principal component analysis [TB99] yields standard PCA; and as we will show in the experimental section of this chapter, the Riemannian L-BFGS method converges quickly, in terms of number of data passes, for leading eigenvector computations with unit norm constrains, which is unconstrained on the sphere manifold as mentioned earlier. This opens up probabilistic PCA on very large datasets as an application area for our algorithm. Our method also applies to cases where PCA has been used for dimensionality reduction in DP-based nonparametric classification models [SN09]. As another example, one can apply these

methods to spectral relaxations of the DP-means objective function [KJ12]; recall that DP-means is an extension of K-means with an unbounded number of clusters, derived from a small-variance asymptotic analysis of Dirichlet Process mixture of Gaussians. Similar to the spectral relaxation of K-means [ZHD$^+$01], one clusters the top eigenvectors of the kernel matrix with eigenvalues larger than the DP-means penalty term, as discussed in §5.1 of [KJ12]; the leading eigenvectors can of course be identified by the algorithm discussed next. As a third example, one could envision learning the spectral decomposition of the covariance matrices in a Gaussian process. Recall that the Gaussian process is uniquely identified by a mean function and a covariance function, and serves as a robust nonparametric prior in function space (for a regression setting); the covariance matrix is the instantiation of the covariance function for the observed set of datapoints, and is usually given by a kernel function. Efficient computation of leading eigenvectors of these matrices is a challenge in the Gaussian process setting, and could potentially be an application area for fast and accurate Riemannian optimization algorithms like the one proposed here.

Contributions: The main contributions of the work in this chapter may be summarized as follows:

1. We propose a stochastic L-BFGS method for Riemannian manifolds using stochastic variance reduction techniques for the first-order gradient estimates, and analyze the convergence for both convex and nonconvex functions under standard assumptions.

2. Our proof for strongly convex functions is different from those of recently proposed stochastic L-BFGS algorithms using variance-reduced gradients in Euclidean space [MNJ16] due to different bounds on the stochastic gradients. We do not use sectional curvature bounds in our proof for the convex case, making it structurally different from that of Riemannian SVRG [ZRS16].

3. We show strong experimental results on Karcher mean computations and calculation of leading eigenvalues, with noticeably better performance than Riemannian SVRG and VR-PCA; the latter is one of the best performing Euclidean algorithms for the finding dominant eigenvalues, that also uses stochastic variance-reduced gradients.

5.1 Preliminaries

5.1.1 Riemannian geometry

We begin with a brief overview of the differential geometric concepts we use in this work. We consider C^∞ (smooth) manifolds that are locally homeomorphic to open subsets of $\mathbb{R}^D$, in the sense that the neighborhood of each point can be assigned a system of coordinates of appropriate dimensionality. Formally, this is defined with the notion of a *chart* $c : U \to \mathbb{R}^D$ at each $x \in \mathscr{M}$, where $U \subset \mathscr{M}$ is an open subspace containing x. Smooth manifolds are ones with covering collections of differentiable (C^∞) charts. A *Riemannian metric* $g(\cdot,\cdot)$ is a bilinear C^∞ tensor field of type $\binom{0}{2}$, that is also symmetric and positive definite. A manifold endowed with such a metric is called a Riemannian manifold. The tangent space $T_x\mathscr{M}$ at every $x \in \mathscr{M}$ is a vector space, with the Riemannian metric $g : T_x\mathscr{M} \times T_x\mathscr{M} \to \mathbb{R}$ as the attendant metric. g then induces a norm for vectors in the tangent space , which we denote by $\|\cdot\|$.

Riemannian manifolds are endowed with the Levi-Civita connection, which induces the notion of parallel transport of vectors from one tangent space to another along a geodesic, in a metric preserving way. That is, we have an operator $\Gamma_\gamma : T_x\mathscr{M} \to T_y\mathscr{M}$ where, informally speaking, γ joins x and y, and for any $u, v \in T_x\mathscr{M}$, we have $g(u,v) = g(\Gamma(u),\Gamma(v))$. The parallel transport can be shown to be an isometry.

For every smooth curve $\gamma : [0,1] \to \mathscr{M}$ lying in $\mathscr{M}$, we denote its velocity vector as $\dot\gamma(t) \in T_x\mathscr{M}$ for each $t \in [0,1]$, with the "speed" given by $\|\dot\gamma(t)\|$. The length of such a curve

is usually measured as $L(\gamma) = \int_0^1 \|\dot{\gamma}(t)\| dt$. Denoting the covariant derivative along γ of some $v \in T_x\mathscr{M}$, with respect to the Riemannian (Levi-Civita) connection by Av, we call $A\dot{\gamma}$ the *acceleration* of the curve. Curves with constant velocities ($A\dot{\gamma} \equiv 0$) are called *geodesics*, and can be shown to generalize the notion of Euclidean straight lines. We assume that every pair $x, y \in \mathscr{M}$ can be connected by a geodesic γ s.t. $\gamma(0) = x$ and $\gamma(1) = y$. Immediately we have the notion of "distance" between any $x, y \in \mathscr{M}$ as the minimum length of all geodesics connecting x and y, assuming the manifolds are *geodesically complete* as mentioned above, in that every countable decreasing sequence of lengths of geodesics connecting a pair of points has a well-defined limit.

The geodesic induces a useful operator called the *exponential map*, defined as $\text{Exp}_x : T_x\mathscr{M} \to \mathscr{M}$ s.t. $\text{Exp}_x(v) = \gamma(1)$ where $\gamma(0) = x, \gamma(1) = y$ and $\dot{\gamma}(0) = v$. If there is a unique geodesic connecting x and y, then the exponential map has an inverse, denoted by $\text{Exp}_x^{-1}(y)$. The length of this geodesic can therefore be seen to be $\|\text{Exp}_x^{-1}(y)\|$.

The derivative D of a differentiable function is defined using the Riemannian connection by the following equivalence: $Df(x)v = vf$, where $v \in T_x\mathscr{M}$. Then, by the Riesz representation theorem, there exists a gradient $\nabla f(x) \in T_x\mathscr{M}$ s.t. $\forall v \in T_x\mathscr{M}$, $Df(x)v = g_x(\nabla f(x), v)$. Similarly, we can denote the Hessian as follows: $D^2 f(x)(\cdot,\cdot) : T_x\mathscr{M} \times T_x\mathscr{M} \to \mathbb{R}$. We denote the mapping from $v \in T_x\mathscr{M}$ to the Riesz representation of $D^2 f(x)(v,\cdot)$ by $\nabla^2 f(x)$. One can consult standard textbooks on differential geometry [Lee97, Boo86] for more details.

5.1.2 Convexity and Lipschitz smoothness on manifolds

Similar to [ZS16, SH15, ZRS16], we define manifold (or geodesic) convexity concepts analogous to the Euclidean baselines, as follows : a set $U \subset \mathscr{M}$ is convex on the manifold if $\forall x, y \in U$ there exists a geodesic γ connecting x, y that completely lies in U, i.e. $\gamma(0) =$

$x, \gamma(1) = y$. Then, a function can be defined as convex w.r.t. geodesics if $\forall x, y \in U$ where $\exists \gamma$ connecting x, y on the manifold, we have:

$$f(\gamma(t)) \leq tf(x) + (1-t)f(y) \, \forall t \in [0,1].$$

We can also define a notion of strong convexity as follows: a function f is called $S-$strongly convex if for any $x, y \in U$ and (sub)gradient ∇_x, we have

$$f(y) \geq f(x) + g_x(\nabla_x, \text{Exp}_x^{-1}(y)) + \frac{S}{2}\|\text{Exp}_x^{-1}(y)\|^2. \tag{5.1}$$

We define Lipschitz smoothness of a function f by imposing Lipschitz continuity on the gradients, as follows: $\forall x, y \in U$,

$$\|\nabla(x) - \Gamma_\gamma \nabla(y)\| \leq L\|\text{Exp}_x^{-1}(y)\|,$$

where L is the smoothness parameter. Analogous to the Euclidean case, this property can also be formulated as::

$$f(y) \leq f(x) + g_x(\nabla_x, \text{Exp}_x^{-1}(y)) + \frac{L}{2}\|\text{Exp}_x^{-1}(y)\|^2. \tag{5.2}$$

5.2 Stochastic Riemannian L-BFGS

In this section we present our stochastic variance-reduced L-BFGS algorithm on Riemannian manifolds and analyze the convergence behavior for convex and nonconvex differentiable functions on Riemannian manifolds. We assume these manifolds to be L-Lipschitz smooth, as defined above, with existence of unique distance-minimizing geodesics between every two points, i.e. our manifolds are geodesically complete; this allows us to have a well-defined inverse exponential map that encodes the distance between a pair of points on the manifold. For the convergence analysis, we also assume f to have a unique minimum at $x^* \in U$, where U is a compact convex subset of the manifold.

5.2.1 The Algorithm

The pseudocode is shown in Algorithm 2. We provide a brief discussion of the salient properties, and compare it to similar algorithms in the Euclidean domain, for example [BHNS14, MNJ16], as well as those on Riemannian manifolds, for example [SH15]. To begin, note that ∇ denotes the Riesz representation of the gradient D, as defined in §5.1.1. We denote full gradients by ∇ and stochastic gradients by $\tilde{\nabla}$. Similar to other stochastic algorithms with variance-reduction, we use two loops: each iteration of the inner loop corresponds to drawing one minibatch from the data and performing the stochastic gradient computations (Steps 10, 11), whereas each outer loop iteration corresponds to two passes over the full dataset, once to compute the full gradient (Step 6) and the other to make multiple minibatch runs (Steps 8 through 30). Compared to the Euclidean setting, note that the computation of the variance-reduced gradient in Step 11 involves an extra step: the gradients ($\nabla f(x)$-s) reside in the tangent spaces of the respective iterates, therefore we have to perform parallel transport to bring them to the same tangent space before performing linear combinations.

To avoid memory issues and complications arising from Hessian-vector computations in the Riemann setting, we chose to update the second correction variable y_r using a simple difference of gradients approximation: $y_r = \tilde{\nabla} f(u_r) - \Gamma_\gamma \tilde{\nabla} f(u_{r-1})$. We should note here that the parallel transport parametrization should be clear from the context; Γ_γ here denotes transporting the vector $\tilde{\nabla} f(u_{r-1}) \in T_{u_{r-1}}\mathscr{M}$ to $T_{u_r}\mathscr{M}$ along the connecting geodesic γ. We omit any relevant annotations from the transport symbol to prevent notational overload. We calculate the first correction pair z_r in one of two ways: (a) as $z_r = \Gamma_\gamma(\eta_2 \rho_{t-1})$, or (b) as $z_r = \Gamma_\gamma\left(-\eta_1 v_{\text{prev}}\right)$. We denote these by **Option 1** and **Option 2** respectively in Alg. 2. Note that in both cases, Γ_γ denotes the parallel transport of the argument to the tangent space at x_t^{s+1}. In our experiments, we noticed faster convergence for the strongly convex centroid

computation problem with **Option 1**, along with computation of the correction pairs every iteration and a low memory pool. For calculating dominating eigenvalues on the unit-radius sphere, **Option 2** yielded better results. Once the corrections pairs z_r, y_r have been computed, we compute the descent step in Step 21 using the standard two-loop recursion formula given in [NW06], using the M correction pairs stored in memory. Note that we use fixed stepsize in the update steps and in computing the correction pairs, and do not impose or perform calculations designed to have them satisfy Armijo or Wolfe conditions.

Compared to the Euclidean algorithms [BHNS14, MNJ16], Alg.2 has some key differences: 1) we did not notice any significant advantage from using separate minibatches in Steps 10 and 18, therefore we use the same minibatch to compute the VR gradient and the correction elements y_r; 2) we do not keep a running average of the iterates for computing the correction element z_r (Steps 15 through 17); 3) we use constant stepsizes throughout the whole process, in contrast to [BHNS14] that uses a decaying sequence. Note that, as seen in Step 24, we use the first-order VR gradient to update the iterates for the first $2R$ iterations; this is because we calculate correction pairs every R steps and evaluate the gradient-inverse Hessian product (Step 26) once at least two pairs have been collected. Similar to [NW06], we drop the oldest pair to maintain the memory depth M. Compared to the algorithms in [SH15, HS15], ours uses stochastic VR gradients, with all the attendant modifications and advantages, and does not use linesearch techniques to satisfy Wolfe conditions.

5.2.2 Analysis of convergence

In this section we provide the main convergence results of the algorithm. We analyze convergence for finite-sum empirical risk minimization problems of the following form:

$$\min_{x \in \mathscr{M}} f(x) = \frac{1}{N} \sum_{i=1}^{N} f_i(x), \tag{5.3}$$

where the Riemannian manifold is denoted by $\mathscr{M}$. Note that the iterates are updated in Algorithm 2 by taking the exponential map of the descent step multiplied by the stepsize, with the descent step computed as the product of the inverse Hessian estimate and the stochastic variance-reduced gradient using the standard two-loop recursion formula. Thus, to bound the optimization error using the iterates, we will need bounds on both the stochastic gradients and the inverse Hessians. As mentioned in [ZRS16], the methods used to derive the former bounds for Euclidean algorithms cannot be ported directly to manifolds due to metric nonlinearities; see the proof of Proposition 1 for details. For the latter, we follow the standard template for L-BFGS algorithms in the literature [RW12, NW06, BHNS14]. To begin, we make the following assumptions:

Assumption 1. The function f in (5.3) is strongly convex on the manifold, whereas the f_is are individually convex.

Assumption 2. There exist $\lambda, \Lambda \in (0,\infty)$, $\lambda < \Lambda$ s.t. $\lambda \|v\|_x^2 \leq D^2 f \leq \Lambda \|v\|_x^2 \quad \forall v \in T_x\mathscr{M}$.

These two assumptions allow us to (a) guarantee that f has a unique minimizer x^* in the convex sublevel set U, and (b) derive bounds on the inverse Hessian updates using BFGS update formulae for the Hessian approximations. Similar to the Euclidean case, these can be written as follows:

$$\hat{B}_r = \Gamma_\gamma \left[\hat{B}_{r-1} - \frac{B_{r-1}(s_{r-1},\cdot)\hat{B}_{r-1}s_{r-1}}{B_{r-1}(s_{r-1},s_{r-1})} \right] \Gamma_\gamma^{-1}, \tag{5.4}$$

and by the Sherman-Morrison-Woodbury lemma, that of the inverse:

$$H_r = \Gamma_\gamma \left[G^{-1} H_{r-1} G + \frac{g_{x_{r-1}}(s_{r-1},\cdot)s_{r-1}}{y_{r-1}s_{r-1}} \right] \Gamma_\gamma^{-1},$$

where $G = I - \frac{g_{x_{r-1}}(s_{r-1},\cdot)\hat{y}_{r-1}}{y_{r-1}s_{r-1}}$, and $\hat{B}_r$ is the Lax-Milgram representation of the Hessian. Details on these constructs can be found in [RW12], in addition to [Lee97, Boo86].

5.2.2.1 Trace and determinant bounds

To start off our convergence discussions for both convex and nonconvex cases, we derive bounds for the trace and determinants of the Hessian approximations, followed by those for their inverses. The techniques used to do so are straightforward ports of the Euclidean originals [NW06], with some minor modifications to account for differential geometric technicalities. Using the assumptions above, we can prove the following bounds [RW12]:

Lemma 1. *Let $B_r^{s+1} = \left(H_r^{s+1}\right)^{-1}$ be the approximation of the Hessian generated by Algorithm 2, and $\hat{B}_r^{s+1}$ and $\hat{H}_r^{s+1}$ be the corresponding Lax-Milgram representations. Let M, the memory parameter, be the number of correction pairs used to update the inverse Hessian approximation. Then, under Assumptions 1 and 2, we have:*

$$\mathrm{tr}(\hat{B}_r^{s+1}) \leq \mathrm{tr}(\hat{B}_0^{s+1}) + M\Lambda, \quad \det(\hat{B}_r^{s+1}) \geq \det(\hat{B}_0^{s+1}) \frac{\lambda^M}{(\mathrm{tr}(\hat{B}_0^{s+1}) + \Lambda M)^M}.$$

Also, $\gamma I \preceq \hat{H}_r^{s+1} \preceq \Gamma I$, for some $\Gamma \geq \gamma > 0$.

From a notational perspective, recall that our notation for the parallel transport operator is Γ_γ, with the subscript denoting the geodesic. The symbols γ and Γ in Lemma 1 above are unrelated to these geometric concepts, merely being the derived bounds on the eigenvalues of inverse Hessian approximations. The proof is given in the supplementary for completeness.

5.2.2.2 Convergence result for strongly convex functions

Our convergence result for strongly convex functions on the manifold can be stated as follows:

Proposition 1. *Let the Assumptions 1 and 2 hold. Further, let the $f(\cdot)$ in* (5.3) *be S-strongly convex, and each of the f_i be L-smooth, as defined earlier. Define the following constants: $p = \left[LS^{-1} + 2\eta_2 S^{-1}\left\{2\eta L^3\Gamma^2 - S\kappa\gamma\right\}\right]$, and $q' = 6\eta^2 L^3\Gamma^2 S^{-1}$. Denote the global optimum*

by x^*. Then the iterate x^{T+1} obtained after T outer loop iterations will satisfy the following condition:

$$\mathbb{E}\left[f(x^{T+1}) - f(x^*)\right] \leq LS^{-1}\beta^T \mathbb{E}\left[f(x^0) - f(x^*)\right],$$

where the constants are chosen to satisfy $\beta = (1-p)^{-1}\left(q' + p^T(1-p-q')\right) < 1$ for linear convergence.

For proving this statement, we will use the L-smoothness (5.2) and S-strong convexity (5.1) conditions mentioned earlier. As in the Euclidean case [JZ13, MNJ16], we will also require a bound on the stochastic variance-reduced gradients. These can be bounded using triangle inequalities and L-smoothness on Riemannian manifolds, as shown in [ZRS16]. This alternative is necessary since the Euclidean bound first derived in [JZ13], using the difference of the objective function at the iterates, cannot be ported directly to manifolds due to metric nonlinearities. Thus we take a different approach in our proof compared to the Euclidean case of [MNJ16], using the interpoint distances defined with the norms of inverse exponential maps. We do not use trigonometric distance inequalities [SH15, Bon13] for the convex case either, making the overall structure different from the proof of Riemannian SVRG as well. The details are deferred to the supplementary due to space limitations. However we do use the trigonometric inequality along with assumed lower bounds on sectional curvature for showing convergence for nonconvex functions, as described next.

5.2.2.3 Convergence for the nonconvex case

Here we provide a convergence result for nonconvex functions satisfying the following condition: $f(x^t) - f(x^*) \leq \kappa^{-1}\|\nabla f(x^t)\|^2$, which automatically holds for strongly convex functions. We assume this to hold even if f is nonconvex, since it allows us to show convergence of the iterates using $\|\nabla f(x^t)\|^2$. Further, similar to [ZS16, ZRS16] we assume that the sectional curvature of the manifold is lower bounded by $c_{\mathcal{S}}$. This allows us to derive

a trigonometric inequality analogous to the Euclidean case, where the sides of the "triangle" are geodesics [Bon13]. The details are given in the supplementary. Additionally, we assume that the eigenvalues of the inverse Hessian are bounded by (γ, Γ) within some suitable region around an optimum. The main result of this section may be stated as follows:

Proposition 2. *Let the sectional curvature of the manifold be bounded below by c_δ, and the f_i be L-smooth. Let x^* be an optimum of $f(\cdot)$ in* (5.3)*. Assume the eigenvalues of the inverse Hessian estimates are bounded. Set $\eta_2 = \mu_0 / \left(\Gamma L n^{\alpha_1} \eta^{\alpha_2}\right)$, $K = mT$, and $m = \lfloor n^{\frac{3\alpha_1}{2}} / \left(3\mu_0 \zeta^{1-2\alpha_2}\right) \rfloor$, where $\alpha_1 \in (0,1]$ and $\alpha_2 \in [0,2]$. Then, for suitable choices of the inverse Hessian bounds γ, Γ, we can find values for the constants $\mu_0 > 0$ and $\varepsilon > 0$ so that the following holds:*

$$\mathbb{E}\|\nabla f(x^T)\|^2 \leq (K\varepsilon)^{-1} L \eta_2^{\alpha_1} \zeta^{\alpha_2} \left(f(x^0) - f(x^*)\right).$$

ζ is defined as $\zeta = \left(\tanh\left(d\sqrt{|c_\delta|}\right)\right)^{-1} d\sqrt{|c_\delta|}$ if $c_\delta < 0$, and 0 otherwise; d is an upper bound on the diameter of the set U mentioned earlier, containing an optimum x^*. The proof is inspired by similar results from both Euclidean [RHS+16] and Riemannian [ZRS16] analyses, and is given in the supplementary. One way to deal with negative curvature in Hessians in Euclidean space is by adding some suitable positive α to the diagonal, ensuring bounds on the eigenvalues. Investigation of such "damping" methods in the Riemannian context could be an interesting area of future work.

Algorithm 2 Riemannian Stochastic VR L-BFGS

Input: Initial value x^0, parameters M and R, learning rates η_1, η_2, minibatch size mb.
Initialize $c = 1$;
Set $r = 0$;
Initialize H_0;
for $t = 0, 1, \ldots$ **do**
  Set $x_0^{t+1} = x^s$;
  Compute full gradient $g^{t+1} = N^{-1} \sum_{i=1}^{N} \nabla f_i(x^t)$;
  for $i = 0, 1, \ldots m-1$ **do**
    Sample minibatch $I_{i,mb} \subset 1, \ldots, N$;
    Compute $\tilde{\nabla} f(x_i^{i+1})$ and $\tilde{\nabla} f(x^i)$ using $I_{i,mb}$;
    Set $v_i^{t+1} = \tilde{\nabla} f(x_i^{t+1}) - \Gamma_\gamma(\tilde{\nabla} f(x^t) - g^{t+1})$;
    if $c \equiv 0 \mod R$ **then**
      Set $r = r + 1$;
      if $r \geq 2$ **then**
        Set $u_r^{t+1} = x_i^{t+1}$;
        Option 1: Compute $z_r^{t+1} = \Gamma_\gamma\left(\eta_2 \rho_{i-1}^{t+1}\right)$;
        Option 2: Compute $z_r^{t+1} = \Gamma_\gamma\left(-\eta_1 v_{\text{prev}}\right)$;
        Compute $y_r^{t+1} = \tilde{\nabla} f(u_r^{t+1}) - \Gamma_\gamma \tilde{\nabla} f(u_{i-1}^{t+1})$ using $I_{i,mb}$;
        Store correction pairs z_r^{t+1} and y_r^{t+1}, using r to maintain memory depth M;
      end if
      Set $x_{\text{prev}} = x_i^{t+1}$, $v_{\text{prev}} = v_i^{t+1}$;
    end if
    if $c < 2R$ **then**
      Set $x_{i+1}^{t+1} = \text{Exp}_{x_i^{t+1}}(-\eta_1 v_i^{t+1})$;
    else
      Compute $\rho_i^{t+1} = H_r^{t+1} v_i^{t+1}$, as mentioned in the text;
      Set $x_{i+1}^{t+1} = \text{Exp}_{x_i^{t+1}}(\eta_2 \rho_i^{t+1})$;
    end if
    Set $c = c + 1$;
  end for
  Set $x^{t+1} = x_m^{t+1}$;
end for

5.3 Experiments

5.3.1 Karcher mean computation for PD matrices

We begin with a synthetic experiment on learning the Karcher mean (centroid) [Bha07] of positive definite matrices. For a collection of matrices $\{\mathbf{W}_i\}_{i=1}^N$, the optimization problem can be stated as follows:

$$\underset{\mathbf{W}\succeq 0}{\operatorname{argmin}} \left\{ \sum_{i=1}^{N} \| \log\left(\mathbf{W}^{-\frac{1}{2}} W_i \mathbf{W}^{-\frac{1}{2}}\right) \|_F^2 \right\}.$$

We compare our minibatched implementation of the Riemannian SVRG algorithm from [ZRS16], denoted as rSVRG, with the stochastic variance-reduced L-BFGS procedure from Algorithm 2, denoted rSV-LBFGS. We implemented both algorithms using the Manopt [BMAS14] and Mixest [HM15] toolkits. We generated three sets of random positive definite matrices, each of size 100×100, with condition numbers $10, 1e2$, and $1e3$, and computed the ground truths using code from [BI13]. Matrix counts were 100 for condition number $1e-2$, and 1000 for the rest. Both algorithms used equal batchsizes of 50 for the first and third datasets, and 5 for the second, and were initialized identically. Both used learning rates satisfying their convergence conditions. In general we found rSV-LBFGS to perform better with frequent correction pair calculations and a low retention rate, ostensibly due to the strong convexity; therefore we used $R = 1$, $M = 2$ for all three datasets. As mentioned earlier, the z_r correction pair was calculated using **Option 1**: $z_r = \Gamma_\gamma(\eta_2 \rho_t)$ where ρ_t is calculated using the two-loop recursion. We used standard retractions to approximate the exponential maps. The retraction formulae for both symmetric PD and sphere manifolds used in the next section are given in the supplementary.

We calculated the error of iterate $\mathbf{W}$ as $\|\mathbf{W} - \mathbf{W}^*\|_F^2$, with $\mathbf{W}^*$ being the ground truth. The log errors are plotted vs the number of data passes in Fig.5.1. Comparing convergence speed in terms of # data passes is often the preferred approach for benchmarking ML algorithms

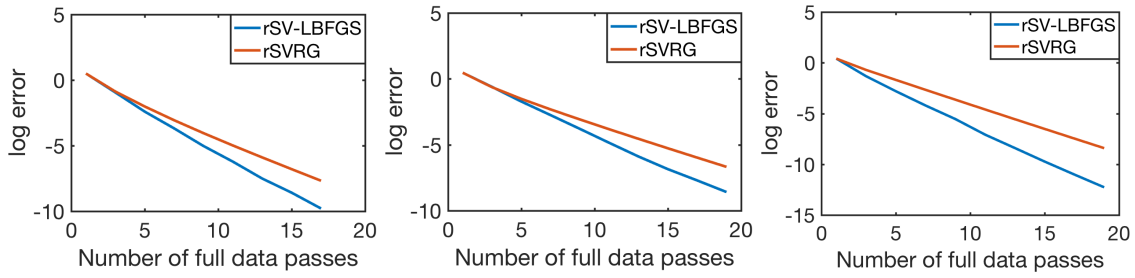


Figure 5.1: Error decay plots for rSV-LBFGS and rSVRG obtained from the three synthetic Karcher mean computation experiments. Figures (a), (b) and (c) show the log-errors for datasets with condition numbers $1e3, 1e2$ and 10 respectively, plotted against number of passes over full dataset for each algorithm. See text for full details.

since it is an implementation-agnostic evaluation and focuses on the key bottleneck (I/O) for big data problems. Comparisons of rSVRG with Riemannian gradient descent methods, both batch and stochastic, can be found in [ZRS16]. From Fig.5.1, we find rSV-LBFGS to converge faster than rSVRG for all three datasets.

5.3.2 Leading eigenvalue computation

Next we conduct a synthetic experiment on calculating the leading eigenvalue of matrices. This of course is a common problem in machine learning, and as such is a unit-norm constrained nonconvex optimization problem in Euclidean space. It can be written as:

$$\min_{\mathbf{z}\in\mathbb{R}^d:\mathbf{z}^T\mathbf{z}=1} -\mathbf{z}^T\left(\frac{1}{N}\sum_{i=1}^{N} d_i d_i^T\right)\mathbf{z},$$

where $D^{d\times N}$ is the data matrix, and d_i are its columns. We can transform this into an unconstrained manifold optimization problem on the sphere defined by the norm constraint. To that end, we generated four sets of datapoints, for eigengaps 0.005, 0.05, 0.01 and 0.1, using the techniques described in [Sha15]. Each dataset contains $100,000$ vectors of dimension 1000. We used a minibatch size of 100 for the two Riemannian algorithms. As before, learning rates for rSVRG were chosen according to the bounds in [ZRS16].

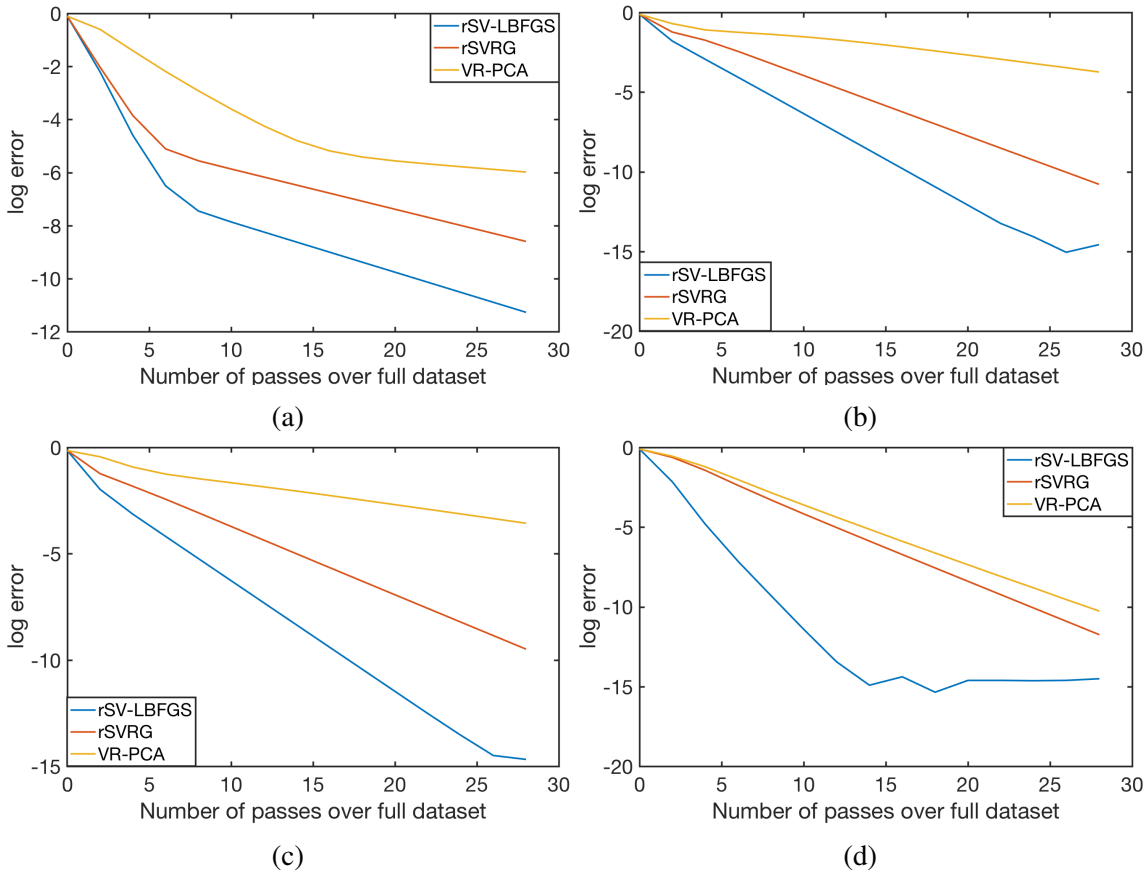


Figure 5.2: Error decay plots for rSV-LBFGS, rSVRG and VR-PCA obtained for dominating eigenvalue computation on four synthetic data matrices. Eigengaps were 0.005, 0.05, 0.01 and 0.1 for (a), (b), (c) and (d) respectively. See text for experimental details.

Selecting appropriate values for the four parameters in rSV-LBFGS (the first and second-order learning rates, L and M) was a nontrivial task; after careful grid searches within the bounds defined by the convergence conditions, we chose $\eta_1 = 0.001, \eta_2 = 0.1$, and $M = 10$ for all four datasets. L was set to 5 for the dataset with eigengap 0.005, and 10 for the rest. The z_r correction pair was calculated using **Option 2**: $z_r = \Gamma_\gamma\left(-\eta_1 v_{\text{prev}}\right)$. We plot the performance of rSV-LBFGS, rSVRG and VR-PCA in Fig.5.2. Extensive comparisons of VR-PCA with other Euclidean algorithms have been conducted in [Sha15]; we do not

repeat them here. We computed the error of iterate $\mathbf{z}$ as $1 - (Ne^*)^{-1} \|D^T\mathbf{z}\|_2^2$, e^* being the ground truth obtained from Matlab's *eigs*.

The VR-PCA algorithm is faster in terms of wall-clock time than our implementations of the Riemannian algorithms: average per-epoch times were 1.04 seconds, 2.723 seconds and 3.211 seconds for VR-PCA, rSVRG, and rSV-LBFGS respectively. However in terms of the number of data passes, we see that the rSV-LBFGS method performs well on all four datasets, reaching errors of the order of $1e-15$ well before VR-PCA and rSVRG in the last three cases. The performance delta relative to VR-PCA is particularly noticeable in each of the four cases; we consider this to be a noteworthy result for fixed-stepsize algorithms on Riemannian manifolds.

5.4 Conclusion

We propose a novel L-BFGS algorithm on Riemannian manifolds with variance reduced stochastic gradients, and provide theoretical analyses for strongly convex functions on manifolds. We conduct experiments on computing Riemannian centroids for symmetric positive definite matrices, and calculation of leading eigenvalues, both using large scale datasets. Our algorithm outperforms other Riemannian optimization algorithms with fixed stepsizes in both cases, and performs noticeably better than one of the fastest stochastic algorithms in Euclidean space, VR-PCA, for the latter case.

5.5 Proof Details

We present the convergence results from Propositions 1 and 2 in the main text in this section.

5.5.1 Convexity and Lipschitz smoothness on manifolds

We begin by recalling the convexity and smoothness definitions on the manifold. Similar to [ZS16, SH15, ZRS16], we define manifold (or geodesic) convexity concepts analogous to the Euclidean baselines, as follows : a set $U \subset \mathscr{M}$ is convex on the manifold if $\forall x, y \in U$ there exists a geodesic γ connecting x, y that completely lies in U, i.e. $\gamma(0) = x, \gamma(1) = y$. Then, a function can be defined as convex w.r.t. geodesics if $\forall x, y \in U$ where $\exists \gamma$ connecting x, y on the manifold, we have:

$$f(\gamma(t)) \leq tf(x) + (1-t)f(y) \, \forall t \in [0,1].$$

We can also define a notion of strong convexity as follows: a function f is called $S-$strongly convex if for any $x, y \in U$ and (sub)gradient ∇_x, we have

$$f(y) \geq f(x) + g_x(\nabla_x, \mathrm{Exp}_x^{-1}(y)) + \frac{S}{2}\|\mathrm{Exp}_x^{-1}(y)\|^2. \tag{5.5}$$

We define Lipschitz smoothness of a function f by imposing Lipschitz continuity on the gradients, as follows: $\forall x, y \in U$,

$$\|\nabla(x) - \Gamma_\gamma \nabla(y)| \leq L\|\mathrm{Exp}_x^{-1}(y)\|,$$

where L is the smoothness parameter. Analogous to the Euclidean case, this property can also be formulated as::

$$f(y) \leq f(x) + g_x(\nabla_x, \mathrm{Exp}_x^{-1}(y)) + \frac{L}{2}\|\mathrm{Exp}_x^{-1}(y)\|^2. \tag{5.6}$$

5.5.2 Analysis of convergence

We analyze convergence for finite-sum empirical risk minimization problems of the following form:

$$\min_{x\in\mathscr{M}} f(x) = \frac{1}{N}\sum_{i=1}^{N} f_i(x), \tag{5.7}$$

where the Riemannian manifold is denoted by $\mathscr{M}$. Note that the iterates are updated in Algorithm 2 by taking the exponential map of the descent step multiplied by the stepsize, with the descent step computed as the product of the inverse Hessian estimate and the stochastic variance-reduced gradient using the standard two-loop recursion formula. Thus, to bound the optimization error using the iterates, we will need bounds on both the stochastic gradients and the inverse Hessians. As mentioned in [ZRS16], the methods used to derive the former bounds for Euclidean algorithms cannot be ported directly to manifolds due to metric nonlinearities; see the proof of Proposition 1 for details. For the latter, we follow the standard template for L-BFGS algorithms in the literature [RW12, NW06, BHNS14]. To begin, we make the following assumptions:

Assumption 1. The function f in (5.7) is strongly convex on the manifold, whereas the f_is are individually convex.

Assumption 2. There exist $\lambda,\Lambda\in(0,\infty)$, $\lambda<\Lambda$ s.t. $\lambda\|v\|_x^2 \le D^2 f \le \Lambda\|v\|_x^2 \quad \forall v \in T_x\mathscr{M}$.

These two assumptions allow us to (a) guarantee that f has a unique minimizer x^* in the convex sublevel set U, and (b) derive bounds on the inverse Hessian updates using BFGS update formulae for the Hessian approximations. Similar to the Euclidean case, these can be written as follows:

$$\hat{B}_r = \Gamma_\gamma \left[\hat{B}_{r-1} - \frac{B_{r-1}(s_{r-1},\cdot)\hat{B}_{r-1}s_{r-1}}{B_{r-1}(s_{r-1},s_{r-1})}\right]\Gamma_\gamma^{-1}, \tag{5.8}$$

and by the Sherman-Morrison-Woodbury lemma, that of the inverse:

$$H_r = \Gamma_\gamma \left[G^{-1} H_{r-1} G + \frac{g_{x_{r-1}}(s_{r-1}, \cdot) s_{r-1}}{y_{r-1} s_{r-1}} \right] \Gamma_\gamma^{-1},$$

where $G = I - \frac{g_{x_{r-1}}(s_{r-1}, \cdot) \hat{y}_{r-1}}{y_{r-1} s_{r-1}}$. The $\hat{B}_r$ is the Lax-Milgram representation of the Hessian [RW12].

5.5.2.1 Trace and determinant bounds

To start off our convergence discussions for both convex and nonconvex cases, we derive bounds for the trace and determinants of the Hessian approximations, followed by those for their inverses. The techniques used to do so are straightforward ports of the Euclidean originals [NW06], with some minor modifications to account for differential geometric technicalities. Using the assumptions above, we can prove the following bounds [RW12]:

Lemma 1. *Let $B_r^{s+1} = \left(H_r^{s+1}\right)^{-1}$ be the approximation of the Hessian generated by Algorithm 2, and $\hat{B}_r^{s+1}$ and $\hat{H}_r^{s+1}$ be the corresponding Lax-Milgram representations. Let M, the memory parameter, be the number of correction pairs used to update the inverse Hessian approximation. Then, under Assumptions 1 and 2, we have:*

$$\mathrm{tr}(\hat{B}_r^{s+1}) \leq \mathrm{tr}(\hat{B}_0^{s+1}) + M\Lambda, \quad \det(\hat{B}_r^{s+1}) \geq \det(\hat{B}_0^{s+1}) \frac{\lambda^M}{(\mathrm{tr}(\hat{B}_0^{s+1}) + \Lambda M)^M}.$$

Also, $\gamma I \preceq \hat{H}_r^{s+1} \preceq \Gamma I$, for some $\Gamma \geq \gamma > 0$.

Proof. For brevity of notation we temporarily drop the $(s+1)$ superscript. The proof for the Euclidean case [BHNS14] can be generalized to the Riemannian scenario in a straightforward way, as follows. Define the average Hessian G_r by

$$G_r(\cdot, \cdot) = \int_0^1 D^2[f(tz_r)](\cdot, \cdot) dt,$$

such that $y_r = G_r(z_r, \cdot)$. Then, it can be easily shown that G_r satisfies the bounds in Assumption 2. Furthermore, we have the following useful inequalities

$$\frac{y_r z_r}{\|z_r\|^2} = \frac{G_r(s_r, s_r)}{\|z_r\|^2} \geq \lambda, \qquad \frac{\|y_r\|^2}{y_r z_r} \leq \Lambda. \tag{5.9}$$

Let $\hat{y}_r$ be the Riesz representation of y_r. Recall that parallel transport is an isometry along the unique geodesics, which implies invariance of the trace operator. Then using the L-BFGS update (5.8) and (5.9), we can bound the trace of the Lax-Milgram representation of the Hessian approximations as follows:

$$\begin{aligned}
\operatorname{tr}(\hat{B}_r) &= \operatorname{tr}(\Gamma_\gamma \hat{B}_{r-1} \Gamma_\gamma^{-1}) - \frac{\|\Gamma_\gamma \hat{B_{r-1}} s_{r-1}\|^2}{B_{r-1}(s_{r-1}, s_{r-1})} + \frac{\|\Gamma_\gamma \hat{y}_{r-1}\|^2}{y_{r-1} s_{r-1}} \\
&\leq \operatorname{tr}(\Gamma_\gamma \hat{B}_{r-1} \Gamma_\gamma^{-1}) + \frac{\|\Gamma_\gamma \hat{y}_{r-1}\|^2}{y_{r-1} s_{r-1}} \\
&\leq \operatorname{tr}(B_0) + M\Lambda.
\end{aligned}$$

This therefore proves boundedness of the largest eigenvalue of the $\hat{B}_r$ estimates.

Similarly, to get a lower bound for the minimum eigenvalue, we bound the determinant as follows:

$$\begin{aligned}
\det(\hat{B}_r) &= \det(\Gamma_\gamma B_{r-1} \Gamma_\gamma^{-1}) \cdot \det\left(I - \frac{\hat{B}_{r-1} s_{r-1} s_{r-1}}{B_{r-1}(s_{r-1}, s_{r-1})} + \hat{B}_{r-1}^{-1} \frac{y_{r-1} y_{r-1}}{y_{r-1} s_{r-1}} \right) \\
&= \det(\Gamma_\gamma B_{r-1} \Gamma_\gamma^{-1}) \frac{y_{r-1} s_{r-1}}{B_{r-1}(s_{r-1}, s_{r-1})} \\
&= \det(\Gamma_\gamma B_{r-1} \Gamma_\gamma^{-1}) \frac{y_{r-1} s_{r-1}}{\|s_{r-1}\|^2} \cdot \frac{\|s_{r-1}\|^2}{B_{r-1}(s_{r-1}, s_{r-1})} \\
&\geq \det(\Gamma_\gamma B_{r-1} \Gamma_\gamma^{-1}) \frac{\lambda}{\lambda_{max}(B_{r-1})},
\end{aligned}$$

where we use $\lambda_{\max}$ to denote the maximum eigenvalue of B_{r-1}, and use (5.9). Since $\lambda_{\max}$ is bounded above by the trace of $(\hat{B}_r)$, we can telescope the inequality above to get

$$\det(\hat{B}_r) \geq \det(B_0) \frac{\lambda^M}{(\operatorname{tr}(B_0) + M\Lambda)^M}.$$

The bounds on the maximum and minimum eigenvalues of B_r thus derived allows us to infer corresponding bounds for those of H_r as well, since by definition $H_r = \hat{B}_r^{-1}$. □

5.5.2.2 Convergence results for the strongly convex case

Next we provide a brief overview of the bounds necessary to prove our convergence result. First, note the following bound implied by the Lipschitz continuity of the gradients:

$$f(x_{t+1}^{s+1}) \leq f(x_t^{s+1}) + g(\nabla f(x_t^{s+1}), \mathrm{Exp}_{x_t^{s+1}}^{-1}(x_{t+1}^{s+1})) + \frac{L}{2}\|\mathrm{Exp}_{x_t^{s+1}}^{-1}(x_{t+1}^{s+1})\|^2.$$

Note the update step fom line 11 of Algorithm 2: $x_{t+1}^{s+1} = \mathrm{Exp}_{x_t^{s+1}}(-\eta H_r^{s+1} v_t^{s+1})$. We can replace the inverse exponential map in the inner product above by the quantity in the parentheses. In order to replace H_r^{s+1} by the eigen-bounds from Lemma 1, we invoke the following result (Lemma 5.8 from [Lee97]:

Lemma 2. *For any $D \in T_x\mathscr{M}$ and $c,t \in \mathbb{R}$, $\gamma_{cD}(t) = \gamma_D(ct)$,*

where v is the "speed" of the geodesic. This allows us to write $\mathrm{Exp}_x(cv) = \gamma_v(c) = \gamma_{cv}(1)$. Recall that for Riemann geodesics we have $\|\dot{\gamma(t)}\| = \bar{s}$ for all $t \in [0,1]$, a constant.

Proposition 1. *Let the Assumptions 1 and 2 hold. Further, let the $f(\cdot)$ in* (5.7) *be S-strongly convex, and each of the f_i be L-smooth, as defined earlier. Define the following constants: $p = \left[LS^{-1} + 2\eta_2 S^{-1}\left\{2\eta L^3\Gamma^2 - S\kappa\gamma\right\}\right]$, and $q' = 6\eta^2 L^3\Gamma^2 S^{-1}$. Denote the global optimum by x^*. Then the iterate x^{T+1} obtained after T outer loop iterations will satisfy the following condition:*

$$\mathbb{E}\left[f(x^{T+1}) - f(x^*)\right] \leq LS^{-1}\beta^T \mathbb{E}\left[f(x^0) - f(x^*)\right],$$

where the constants are chosen to satisfy $\beta = (1-p)^{-1}\left(q' + p^T(1-p-q')\right) < 1$ for linear convergence.

Proof. From the L-smoothness condition (5.6), we have the following:

$$\begin{aligned} f(x_{i+1}^{t+1}) &\leq f(x_i^{t+1}) + g\left(\nabla f(x_i^{t+1}) \cdot \mathrm{Exp}_{x_i^{t+1}}^{-1}(x_{i+1}^{t+1})\right) + \frac{L}{2}\|\mathrm{Exp}_{x_i^{t+1}}^{-1}(x_{i+1}^{t+1})\|^2 \\ &= f(x_i^{t+1}) - \eta_2 \cdot g\left(\nabla f(x_i^{t+1}), H_r^{t+1} v_i^{t+1}\right) + \frac{L\eta_2^2}{2}\|H_r^{t+1} v_i^{t+1}\|^2, \end{aligned}$$

where we have omitted subscripts from the metric. Taking expectations, and using the bounds on the inverse Hessian estimates derived in Lemma 1, we have the following:

$$\mathbb{E}f(x_{i+1}^{t+1}) \leq \mathbb{E}f(x_i^{t+1}) - \eta_2\gamma\|\nabla f(x_i^{t+1})\|^2 + \eta_2^2 L^3\Gamma^2\left[2\|\mathrm{Exp}_{x_i^{t+1}}^{-1}(x^*)\|^2 + 3\|\mathrm{Exp}_{x^t}^{-1}(x^*)\|^2\right], \tag{5.10}$$

where we have used the following bound on the stochastic variance-reduced gradients derived in [ZRS16]:

$$\mathbb{E}\|v_i^{t+1}\|^2 \leq 4L^2\|\mathrm{Exp}_{x_i^{t+1}}^{-1}(x^*)\|^2 + 6L^2\|\mathrm{Exp}_{x^t}^{-1}(x^*)\|^2.$$

This can be derived using triangle inequalities and the L-smoothness assumption. Note that the bound is different from the Euclidean case [JZ13], due to technicalities introduced by the Riemannian metric not being linear in general.

Now, recall the condition $f(x^t) - f(x^*) \leq 2\kappa)^{-1}\|\nabla f(x^t)\|^2$, which follows from strong convexity. Using this, we can derive a bound on the ∇ term in (5.10) as follows:

$$\|\nabla f(x_i^{t+1})\|^2 \geq 2\kappa\left(f(x_i^{t+1}) - f(x^*)\right) \geq S\kappa\|\mathrm{Exp}_{x_i^{t+1}}^{-1}(x^*)\|^2,$$

where the second inequality follows from S-strong convexity (5.5), since $\nabla f(x^*) = 0$. Plugging this into (5.10), we have the following:

$$\begin{aligned}\mathbb{E}f(x_{i+1}^{t+1}) \leq f(x_i^{t+1}) + \eta_2\left[2\eta L^3\Gamma^2 - S\kappa\gamma\right]\|\mathrm{Exp}_{x_i^{t+1}}^{-1}(x^*)\|^2 \\ + 3\eta_2^2 L^3\Gamma^2\|\mathrm{Exp}_{x^t}^{-1}(x^*)\|^2.\end{aligned} \tag{5.11}$$

Now, note that S-strong convexity allows us to write the following:

$$\begin{aligned}\frac{S}{2}\|\mathrm{Exp}_{x_{i+1}^{t+1}}^{-1}(x^*)\|^2 &\leq f(x_{i+1}^{t+1}) - f(x^*) \\ &= \left[f(x_{i+1}^{t+1}) - f(x_i^{t+1})\right] + \left[f(x_i^{t+1}) - f(x^*)\right] \\ &\leq \left[f(x_{i+1}^{t+1}) - f(x_i^{t+1})\right] + \frac{L}{2}\|\mathrm{Exp}_{x_i^{t+1}}^{-1}(x^*)\|^2,\end{aligned}$$

where the last step follows from L-smoothness. Taking expectations of both sides, and using (5.11) for the first component on the right, we have

$$\begin{aligned}\mathbb{E}\|\mathrm{Exp}_{x_{i+1}^{t+1}}^{-1}(x^*)\|^2 \leq \left[\frac{L}{S} + \frac{2\eta_2}{S}\left\{2\eta L^3\Gamma^2 - S\kappa\gamma\right\}\right]\|\mathrm{Exp}_{x_{i+1}^{t+1}}^{-1}(x^*)\|^2 \\ + \frac{6\eta_2^2 L^3\Gamma^2}{S}\|\mathrm{Exp}_{x^t}^{-1}(x^*)\|^2.\end{aligned} \tag{5.12}$$

Now, we denote $p = \left[\frac{L}{S} + \frac{2\eta_2}{S}\left\{2\eta L^3\Gamma^2 - S\kappa\gamma\right\}\right]$, and $q' = \frac{6\eta_2^2 L^3\Gamma^2}{S}$. Then, taking expectations over the sigma algebra of all the random variables till minibatch m, and some algebra, it can be shown that:

$$\mathbb{E}\|\mathrm{Exp}_{x_m^{t+1}}^{-1}(x^*)\|^2 - q\mathbb{E}\|\mathrm{Exp}_{x_0^{t+1}}^{-1}(x^*)\|^2 \leq p^m(1-q)\|\mathrm{Exp}_{x_0^{t+1}}^{-1}(x^*)\|^2,$$

where $q = (1-p)^{-1}q'$. Note that this provides a bound on the iterate at the end of the inner minibatch loop. Telescoping further, we have the bound

$$\mathbb{E}\|\mathrm{Exp}_{x^{T+1}}^{-1}(x^*)\|^2 \leq \beta^T\mathbb{E}\|\mathrm{Exp}_{x^0}^{-1}(x^*)\|^2,$$

where $\beta = \frac{q'+p^T(1-p-q')}{1-p}$. Then, using this result with a final appeal to the L-Lipschitz and S-strong convexity conditions, we have the bounds

$$\begin{aligned}\mathbb{E}\left[f(x^{T+1}) - f(x^*)\right] &\leq \frac{L}{2}\mathbb{E}\|\mathrm{Exp}_{x^{T+1}}^{-1}(x^*)\|^2 \\ &\leq \frac{L}{S}\beta^T\left[f(x^0) - f(x^*)\right],\end{aligned}$$

thereby completing the proof. □

5.5.2.3 Convergence results for the nonconvex case

We begin with the following inequality involving the side lengths of a geodesic "triangle" [SH15, Bon13]:

Lemma 3. *Let the sectional curvature of a Riemannian manifold be bounded below by c_δ. Let A be the angle between sides of length b and c in a triangle on the manifold, with the*

third side of length a, as usual. Then the following holds:

$$a^2 \leq \frac{c\sqrt{|c_\delta|}}{\tanh\left(c\sqrt{|c_\delta|}\right)}b^2 + c^2 - 2bc\cos A.$$

The cosine is defined using inner products, as in the Euclidean case, and the distances using inverse exponential maps, as seen above. The following sequence of results and proofs are inspired by the basic structure of [RHS+16], with suitable modifications involving the inverse Hessian estimates from the L-BFGS updates.

Lemma 4. *Let the assumptions of Proposition 2 hold. Define the following functions:*

$$c_i = c_{i+1}\left(1 + \beta\eta_2\Gamma + 2\zeta L^2\eta_2^2\Gamma^2\right) + L^3\eta_2^2\Gamma^2,$$
$$\delta_i = \eta_2\gamma - \frac{c_{i+1}\eta_2\Gamma}{\beta} - L\eta_2^2\Gamma - 2c_{i+1}\zeta\eta_2^2\Gamma^2 > 0,$$

where $c_i, c_{i+1}, \beta, \eta_2 > 0$. *Further, for* $0 \leq t \leq T-1$, *define the Lyapunov function* $R_i^{t+1} = \mathbb{E}\left[f(x_i^{t+1}) + c_i\|Exp_{x^t}^{-1}(x_i^{t+1})\|^2\right]$. *Then we have the following bound:*

$$\mathbb{E}\|\nabla f(x_i^{t+1})\|^2 \leq \frac{R_i^{t+1} - R_{i+1}^{t+1}}{\delta_i}.$$

Proof. As with the proof of Proposition 1, we begin with the following bound derived from L-smoothness:

$$\mathbb{E}f(x_{i+1}^{t+1}) \leq \mathbb{E}\left[f(x_i^{t+1}) - \eta_2\gamma\|\nabla f(x_i^{t+1})\|^2 + \frac{L\eta_2^2\Gamma^2}{2}\|v_i^{t+1}\|^2\right],$$

where we have bound the bounds on the inverse Hessian derived earlier. Then using Lemma 3 above, we have:

$$\begin{aligned}
\mathbb{E}\|\mathrm{Exp}_{x^t}^{-1}(x_{i+1}^{t+1})\|^2 &\leq \mathbb{E}\|\mathrm{Exp}_{x^t}^{-1}(x_i^{t+1})\|^2 + \zeta\|\mathrm{Exp}_{x_i^{t+1}}^{-1}(x_{i+1}^{t+1})\|^2 \\
&\quad - 2g\left(\mathrm{Exp}_{x_i^{t+1}}^{-1}(x_{i+1}^{t+1}), \mathrm{Exp}_{x_i^{t+1}}^{-1}(x^*)\right) \\
&\leq \mathbb{E}\left[\|\mathrm{Exp}_{x^t}^{-1}(x_i^{t+1})\|^2 + \zeta\eta_2^2\Gamma^2\|v_i^{t+1}\|^2\right] \\
&\quad + 2\eta_2\Gamma\left[(2\beta)^{-1}\|\nabla f(x_i^{t+1})\|^2 + \frac{\beta}{2}\|\mathrm{Exp}_{x^t}^{-1}(x_i^{t+1})\|^2\right],
\end{aligned}$$

where we have used $g(a,b) \leq \frac{1}{2\beta}\|a\|^2 + \frac{\beta}{2}\|b\|^2$. Note that we have used the norm of the inverse exponential maps as the side lengths in Lemma 3. Using these last two results, we can derive the following bound for the Lyapunov functions R_{i+1}^{t+1}:

$$R_{i+1}^{t+1} \leq \mathbb{E}\left[f(x_i^{t+1}) - \left\{\eta_2\gamma - \frac{c_{i+1}\eta_2\Gamma}{\beta}\right\}\|\nabla f(x_i^{t+1})\|^2\right] + \Gamma^2\left\{c_{i+1}\zeta\eta_2^2 + \frac{L\eta_2^2}{2}\right\}\mathbb{E}\|v_i^{t+1}\|^2 + c_{i+1}\{1+\eta_2\Gamma\beta\}\mathbb{E}\|\mathrm{Exp}_{x^t}^{-1}(x_i^{t+1})\|^2.$$

The norm of the stochastic variance reduced gradient can be bounded as follows [ZRS16, RHS+16]:

$$\mathbb{E}\|v_i^{t+1}\|^2 \leq 2L^2\mathbb{E}\|\mathrm{Exp}_{x^t}^{-1}(x_i^{t+1})\|^2 + 2\mathbb{E}\|\nabla f(x_i^{t+1})\|^2.$$

This allows us to bound the Lyapunov function above as:

$$R_{i+1}^{t+1} \leq R_i^{t+1} - \left\{\eta_2\gamma - \frac{c_{i+1}\eta_2\Gamma}{\beta} - L\eta_2^2\Gamma^2 - 2c_{i+1}\zeta\eta_2^2\Gamma^2\right\}\mathbb{E}\|\nabla f(x_i^{t+1})\|^2,$$

which completes the proof. □

Next we present a bound on $\|\nabla f(\cdot)\|^2$ using the δ_i's defined above (Thm 6 of [ZRS16]):

Lemma 5. *Let the conditions of Lemma 4 hold, and define the quantities therein. Let* $\delta_i > 0$ $\forall i \in [0,m]$*, and* $c_m = 0$*. Let* $\delta_\delta = \min_i \delta_i$*, and* $K = mT$*. Then if we randomly return one of the iterates* $\{x_i^{t+1}\}_{i=1}^m$ *as* x^{t+1}*, then:*

$$\mathbb{E}\|\nabla f(x^t)\|^2 \leq \frac{f(x^0) - f(x^*)}{K\delta_\delta}.$$

This result can be shown by telescoping the bound derived in the previous lemma for the Lyapunov functions, using $c_m = 0$.

Proposition 2. *Let the sectional curvature of the manifold be bounded below by* c_δ*, and the* f_i *be L-smooth. Let* x^* *be an optimum of* $f(\cdot)$ *in* (5.7)*. Assume the eigenvalues of the inverse Hessian estimates are bounded. Set* $\eta_2 = \mu_0/(\Gamma L n^{\alpha_1}\eta^{\alpha_2})$*,* $K = mT$*, and* $m =$

$\lfloor n^{\frac{3\alpha_1}{2}} / \left(3\mu_0 \zeta^{1-2\alpha_2}\right) \rfloor$, *where* $\alpha_1 \in (0,1]$ *and* $\alpha_2 \in [0,2]$. *Then, for suitable choices of the inverse Hessian bounds* γ, Γ*, we can find values for the constants* $\mu_0 > 0$ *and* $\varepsilon > 0$ *so that the following holds:*

$$\mathbb{E}\|\nabla f(x^T)\|^2 \leq (K\varepsilon)^{-1} L \eta_2^{\alpha_1} \zeta^{\alpha_2} \left(f(x^0) - f(x^*)\right).$$

Proof. We define $\beta = L\zeta^{1-\alpha_2} / \left(n^{\frac{\alpha_1}{2}} \Gamma\right)$. Also, as mentioned in the proposition, $\eta_2 = \mu_0 / \left(\Gamma L n^{\alpha_1} \eta^{\alpha_2}\right)$, with appropriate α_1, α_2. Note that we need a bound for δ_δ to plug in the denominator of the bound in Lemma 5 above. This quantity can be lower bounded as follow:

$$\begin{aligned}
\delta_\delta &= \min_i \delta_i \\
&= \min_i \left\{ \eta_2 \gamma - \frac{c_{i+1} \eta_2 \Gamma}{\beta} - L\eta_2^2 \Gamma^2 - 2c_{i+1} \zeta^2 \eta_2^2 \Gamma^2 \right\} \\
&\geq \left\{ \eta_2 \gamma - \frac{c_0 \eta_2 \Gamma}{\beta} - L\eta^2 \Gamma^2 - 2c_0 \zeta \eta_2^2 \Gamma^2 \right\}.
\end{aligned}$$

Now we need to bound c_0. To that end, telescoping the c_{i+1} function defined in Lemma 4 above with $c_m = 0$, and denoting $\theta = \eta_2 \beta \Gamma + 2\zeta \eta_2^2 L^2 \Gamma^2$, we get the following:

$$c_0 = \frac{L\mu_0^2 \{(1+\theta)^m - 1\}}{n^{2\alpha_1} \zeta^{2\alpha_2} \theta}.$$

Using the definitions of η_2 and β above, we note that $\theta < 1/m$, implying $c_0 \leq \frac{L\mu_0}{\zeta n^{\frac{\alpha_1}{2}}}(e-1)$. Plugging this in the bound above, we posit that δ_δ can be bounded below as follows:

$$\begin{aligned}
\delta_\delta &\geq \eta_2 \left\{ \gamma - \frac{\mu_0 \Gamma (e-1)}{\zeta^{2-\alpha_2}} - \frac{\mu_0}{n^{\alpha_2} \zeta^{\alpha_2}} - \frac{2\mu_0^2 (e-1)}{n^{\frac{3\alpha_1}{2}} \zeta^{\alpha_2}} \right\} \\
&\geq \frac{\varepsilon}{L n^{\alpha_1} \zeta^{\alpha_2}},
\end{aligned}$$

for some sufficiently small ε, and suitable choices of the inverse Hessian bounds γ, Γ and the rest of the parameters. Using this bound in the denominator of the right hand side of Lemma 5 above completes the proof. □

5.5.3 Retractions

We approximated the exponential maps with retractions from the Manopt [BMAS14] toolbox. We used the following formulae: $\mathbb{R}_x(\eta\rho) = x\cos\|\eta\rho\|_F + \frac{\eta\rho}{\|\eta\rho\|_F}\sin\|\eta\rho\|_F$ for the sphere manifold, and $\mathbb{R}_x(\eta\rho) = x \cdot M_x(x \setminus \eta\rho)$ for the manifold of symmetric PD matrices, where M_x denotes the matrix exponential, and $\setminus$ is matrix division. Here $x \in \mathscr{M}$ is some point on the manifold, $\rho \in T_x\mathscr{M}$ is some descent step evaluated at x, and η is the stepsize.

Chapter 6: Adaptive Bayesian Sampling with Monte Carlo EM

Markov Chain Monte Carlo sampling is a well-known set of techniques for learning complex Bayesian probabilistic models that arise in machine learning. Typically used in cases where computing the posterior distributions of parameters in closed form is not feasible, MCMC techniques that converge reliably to the target distributions offer a provably correct way (in an asymptotic sense) to draw samples of target parameters from arbitrarily complex probability distributions. A recently proposed method in this domain is Hamiltonian Monte Carlo (HMC) [Nea10, DKPR87], that formulates the target density as an "energy function" augmented with auxiliary "momentum" parameters, and uses discretized Hamiltonian dynamics to sample the parameters while preserving the energy function. The resulting samplers perform noticeably better than random walk-based methods in terms of sampling efficiency and accuracy [Nea10, GC11]. For use in stochastic settings, where one uses random minibatches of the data to calculate the gradients of likelihoods for better scalability, researchers have used Fokker-Planck correction steps to preserve the energy in the face of stochastic noise [CCG14], as well as used auxiliary "thermostat" variables to control the effect of this noise on the momentum terms [DFB$^+$14, RKP16]. As with the batch setting, these methods have exploited energy-preserving dynamics to sample more efficiently than random walk-based stochastic samplers [CCG14, WT11, PT13].

A primary (hyper-)parameter of interest in these augmented energy function-based samplers in the "mass" matrix of the kinetic energy term; as noted by various researchers

[Nea10, GC11, RKP16, PT13, BLL99], this matrix plays an important role in the trajectories taken by the samplers in the parameter space of interest, thereby affecting the overall efficiency. While prior efforts have set this to the identity matrix or some other pre-calculated value [CCG14, DFB$^+$14, WT11], recent work has shown that there are significant gains to be had in efficiency as well as convergent accuracy by reformulating the mass in terms of the target parameters to be sampled [GC11, RKP16, PT13], thereby making the sampler sensitive to the underlying geometry. This is done by imposing a positive definite constraint on the adaptive mass, and using it as the metric of the Riemannian manifold of probability distributions parametrized by the target parameters. This constraint also satisfies the condition that the momenta be sampled from a Gaussian with the mass as the covariance. Often called Riemannian preconditioning, this idea has been applied in both batch [GC11] as well as stochastic settings [RKP16, PT13] to derive HMC-based samplers that adaptively learn the critically important mass matrix from the data.

Although robust, these reformulations often lead to significant complexities in the resultant dynamics; one can end up solving an implicit system of equations in each half-step of the leapfrog dynamics [GC11, RKP16], along with inverting large $\left(O(D^2)\right)$ matrices. This is sometimes sidestepped by performing fixed point updates at the cost of additional error, or restricting oneself to simpler formulations that honor the symmetric positive definite constraint, such as a diagonal matrix [PT13]. While this latter choice ameliorates a lot of the added complexity, it is clearly suboptimal in the context of adapting to the underlying geometry of the parameter space. Thus we would ideally need a mechanism to robustly learn this critical mass hyperparameter from the data without significantly adding to the computational burden.

We address this issue in this work with the Monte Carlo EM (MCEM) [WT90, BH99, McC97, LC01] framework. An alternative to the venerable EM technique, MCEM is used

to locally optimize maximum likelihood problems where the posterior probabilities required in the E step of EM cannot be computed in closed form. In this work, we perform existing dynamics derived from energy functions in the Monte Carlo E step while holding the mass fixed, and use the stored samples of the momentum term to learn the mass in the M step. We address the important issue of selecting appropriate E-step sampling iterations, using error estimates to gradually increase the sample sizes as the Markov chain progresses towards convergence. Combined with an online method to update the mass using sample covariance estimates in the M step, this gives a clean and scalable adaptive sampling algorithm that performs favorably compared to the Riemannian samplers. In both our synthetic experiments and a high dimensional topic modeling problem with a complex Bayesian nonparametric construction [RK15], our samplers match or beat the Riemannian variants in sampling efficiency and accuracy, while being close to an order of magnitude faster.

6.1 Preliminaries

6.1.1 MCMC with Energy-Preserving Dynamics

In Hamiltonian Monte Carlo, the energy function is written as

$$H(\boldsymbol{\theta}, \mathbf{p}) = -\mathscr{L}(\boldsymbol{\theta}) + \frac{1}{2}\mathbf{p}^T M^{-1}\mathbf{p}. \tag{6.1}$$

Here $\mathbf{X}$ is the observed data, and $\boldsymbol{\theta}$ denotes the model parameters. $\mathscr{L}(\boldsymbol{\theta}) = \log p(\mathbf{X}|\boldsymbol{\theta}) + \log p(\boldsymbol{\theta})$ denotes the log likelihood of the data given the parameters along with the Bayesian prior, and $\mathbf{p}$ denotes the auxiliary "momentum" mentioned above. Note that the second term in the energy function, the kinetic energy, is simply the kernel of a Gaussian with the mass matrix M acting as covariance. Hamilton's equations of motions are then applied to this energy function to derive the following differential equations, with the dot accent denoting a

time derivative:

$$\dot{\theta} = M^{-1}\mathbf{p}, \quad \dot{\mathbf{p}} = \nabla \mathscr{L}(\boldsymbol{\theta}).$$

These are discretized using the generalized leapfrog algorithm [Nea10, LR05] to create a sampler that is both symplectic and time-reversible, upto a discretization error that is quadratic in the stepsize.

Machine learning applications typically see the use of very large datasets for which computing the gradients of the likelihoods in every leapfrog step followed by a Metropolis-Hastings correction ratio is prohibitively expensive. To address this, one uses random "minibatches" of the dataset in each iteration [RM51], allowing some stochastic noise for improved scalability, and removes the Metropolis-Hastings (M-H) correction steps [CCG14, WT11]. To preserve the system energy in this context one has to additionally apply Fokker-Planck corrections to the dynamics [YA06]. The stochastic sampler in [CCG14] uses these techniques to preserve the canonical Gibbs energy above (6.1). Researchers have also used the notion of "thermostats" from the molecular dynamics literature [BLL99, FS01, LM15, Hoo85] to further control the behavior of the momentum terms in the face of stochastic noise; the resulting algorithm [DFB$^+$14] preserves an energy of its own [JL11] as well.

6.1.2 Adaptive MCMC using Riemannian Manifolds

As mentioned above, learning the mass matrices in these MCMC systems is an important challenge. Researchers have traditionally used Riemannian manifold refomulations to address this, and integrate the updating of the mass into the sampling steps. In [GC11] the authors use this approach to derive adaptive variants of first-order Langevin dynamics as well as HMC. For the latter the reformulated energy function can be written as:

$$H_{gc}(\boldsymbol{\theta}, \mathbf{p}) = -\mathscr{L}(\boldsymbol{\theta}) + \frac{1}{2}\mathbf{p}^T \mathbf{G}(\boldsymbol{\theta})^{-1}\mathbf{p} + \frac{1}{2}\log\left\{(2\pi)^D |\mathbf{G}(\boldsymbol{\theta})|\right\}, \tag{6.2}$$

where D is the dimensionality of the parameter space. Note that the momentum variable $\mathbf{p}$ can be integrated out to recover the desired marginal density of $\boldsymbol{\theta}$, in spite of the covariance being a function of $\boldsymbol{\theta}$. In the machine learning literature, the authors of [PT13] used a diagonal $\mathbf{G}(\boldsymbol{\theta})$ to produce an adaptive variant of the algorithm in [WT11], whereas the authors in [RKP16] derived deterministic and stochastic algorithms from a Riemannian variant of the Nosé-Poincaré energy [BLL99], with the resulting adaptive samplers preserving symplecticness as well as canonical system temperature.

6.1.3 Monte Carlo EM

The EM algorithm [DLR77] is widely used to learn maximum likelihood parameter estimates for complex probabilistic models. In cases where the expectations of the likelihoods required in the E step are not tractable, one can use Monte Carlo simulations of the posterior instead. The resulting Monte Carlo EM (MCEM) framework [WT90] has been widely studied in the statistics literature, with various techniques developed to efficiently draw samples and estimate Monte Carlo errors in the E step [BH99, McC97, LC01]. For instance, the expected log-likelihood is usually replaced with the following Monte Carlo approximation: $Q(\boldsymbol{\theta}|\boldsymbol{\theta}^t) = \frac{1}{m}\sum_{l=1}^{m} \log p(\mathbf{X}, \mathbf{u}_l^t|\boldsymbol{\theta})$, where $\mathbf{u}$ represents the latent augmentation variables used in EM, and m is the number of samples taken in the E step. While applying this framework, one typically has to carefully tune the number of samples gathered in the E step, since the potential distance from the stationary distribution in the early phases would necessitate drawing relatively fewer samples, and progressively more as the sampler nears convergence.

In this work we leverage this MCEM framework to learn M in (6.1) and similar energies using samples of $\mathbf{p}$; the discretized dynamics constitute the E step of the MCEM framework, with suitable updates to M performed in the corresponding M step. We also use a novel

mechanism to dynamically adjust the sample count by using sampling errors estimated from the gathered samples, as described next.

6.2 Mass-Adaptive Sampling with Monte Carlo EM

6.2.1 The Basic Framework

Riemannian samplers start off by reformulating the energy function, making the mass a function of $\boldsymbol{\theta}$ and adding suitable terms to ensure constancy of the marginal distributions. Our approach is fundamentally different: we cast the task of learning the mass as a maximum likelihood problem over the space of symmetric positive definite matrices. For instance, we can construct the following problem for standard HMC:

$$\max_{M \succ 0} \quad \mathscr{L}(\boldsymbol{\theta}) - \frac{1}{2}\mathbf{p}^T M^{-1}\mathbf{p} - \frac{1}{2}\log|M|. \tag{6.3}$$

Recall that the joint likelihood is $p(\boldsymbol{\theta},\mathbf{p}) \propto \exp(-H(\boldsymbol{\theta},\mathbf{p}))$, $H(\cdot,\cdot)$ being the energy from (6.1). Then, we use correct samplers that preserve the desired densities in the E step of a Monte Carlo EM (MCEM) framework, and use the obtained samples of $\mathbf{p}$ in the corresponding M step to perform suitable updates for the mass M. Specifically, to wrap the standard HMC sampler in our framework, we perform the generalized leapfrog steps [Nea10, LR05] to obtain proposal updates for $\boldsymbol{\theta},\mathbf{p}$ followed by Metropolis-Hastings corrections in the E step, and use the obtained $\mathbf{p}$ values in the M step. The resultant adaptive sampling method is shown in Alg. 3.

Note that this framework can also be applied to stochastic samplers that preserve the energy, upto standard discretization errors. We can wrap the SGHMC sampler [CCG14] in our framework as well, since it uses Fokker-Planck corrections to approximately preserve the energy (6.1) in the presence of stochastic noise. We call the resulting method SGHMC-EM, and specify it in Alg. 3 in the supplementary.

As another example, the SGNHT sampler [DFB$^+$14] is known to preserve a modified Gibbs energy [JL11]; therefore we can propose the following max-likelihood problem for learning the mass:

$$\max_{M \succ 0} \quad \mathscr{L}(\boldsymbol{\theta}) - \frac{1}{2}\mathbf{p}^T M^{-1}\mathbf{p} - \frac{1}{2}\log|M| + \mu(\xi - \bar{\xi})^2/2, \tag{6.4}$$

where ξ is the thermostat variable, and μ, $\bar{\xi}$ are constants chosen to preserve correct marginals. The SGNHT dynamics can used in the E step to maintain the above energy, and we can use the collected $\mathbf{p}$ samples in the M step as before. We call the resultant method SGNHT-EM, as shown in Alg. 4. Note that, unlike standard HMC above, we do not perform Metropolis-Hastings corrections steps on the gathered samples for these cases. As shown in the algorithms, we collect one set of momenta samples per epoch, after the leapfrog iterations. We use *S*_count to denote the number of such samples collected before running an M-step update.

The advantage of this MCEM approach over the parameter-dependent Riemannian variants is twofold:

1. The existing Riemannian adaptive algorithms in the literature [GC11, RKP16, PT13] all start by modifying the energy function, whereas our framework does not have any such requirement. As long as one uses a sampling mechanism that preserves some energy with correct marginals for $\boldsymbol{\theta}$, in a stochastic sense or otherwise, it can be used in the E step of our framework.

2. The primary disadvantage of the Riemannian algorithms is the added complexity in the dynamics derived from the modified energy functions. One typically ends up using generalized leapfrog dynamics [GC11, RKP16], which can lead to implicit systems of equations; to solve these one either has to use standard solvers that have complexity at least cubic in the dimensionality [Dix82, EGG$^+$06], with scalability issues in high dimensional datasets, or use fixed point updates with worsened error guarantees. An alternative approach

is to use diagonal covariance matrices, as mentioned earlier, which ignores the coordinate correlations. Our MCEM approach sidesteps all these issues by keeping the existing dynamics of the desired E step sampler unchanged. As shown in the experiments, we can match or beat the Riemannian samplers in accuracy and efficiency by using suitable sample sizes and M step updates, with significantly improved sampling complexities and runtimes.

6.2.2 Dynamic Updates for the E-step Sample Size

Algorithm 3 HMC-EM

Input: $\boldsymbol{\theta}^{(0)}, \varepsilon, LP\_S, S\_\text{count}$
· Initialize M;
repeat
  · Sample $\mathbf{p}^{(t)} \sim N(0, M)$;
  for $i = 1$ **to** LP_S **do**
    · $\mathbf{p}^{(i)} \leftarrow \mathbf{p}^{(i+\varepsilon-1)}$, $\boldsymbol{\theta}^{(i)} \leftarrow \boldsymbol{\theta}^{(i+\varepsilon-1)}$;
    · $\mathbf{p}^{(i+\frac{\varepsilon}{2})} \leftarrow \mathbf{p}^{(i)} - \frac{\varepsilon}{2}\nabla_{\boldsymbol{\theta}} H(\boldsymbol{\theta}^{(i)}, \mathbf{p}^{(i)})$;
    · $\boldsymbol{\theta}^{(i+\varepsilon)} \leftarrow \boldsymbol{\theta}^{(i)} + \frac{\varepsilon}{2}\nabla_{\mathbf{p}} H(\boldsymbol{\theta}^{(i)}, \mathbf{p}^{(i+\frac{\varepsilon}{2})})$;
    · $\mathbf{p}^{(i+\varepsilon)} \leftarrow \mathbf{p}^{(i+\frac{\varepsilon}{2})} - \frac{\varepsilon}{2}\nabla_{\boldsymbol{\theta}} H(\boldsymbol{\theta}^{(i+\varepsilon)}, \mathbf{p}^{(i+\frac{\varepsilon}{2})})$;
  end for
  · Set $\left(\boldsymbol{\theta}^{(t+1)}, \mathbf{p}^{(t+1)}\right)$ from $\left(\boldsymbol{\theta}^{LP\_S+\varepsilon}, \mathbf{p}^{LP\_S+\varepsilon}\right)$ using Metropolis-Hastings
  · Store MC-EM sample $\mathbf{p}^{(t+1)}$;
  if $(t+1) \bmod S\_\text{count} = 0$ **then**
    · Update M using MC-EM samples;
  end if
  · Update $S_$count as described in the text;
until forever

We now turn our attention to the task of learning the sample size in the E step from the data. The nontriviality of this issue is due to the following reasons: first, we cannot let the sampling dynamics run to convergence in each E step without making the whole process prohibitively slow; second, we have to account for the correlation among successive samples, especially early on in the process when the Markov chain is far from convergence, possibly with "thinning" techniques; and third, we may want to increase the sample count as the chain matures and gets closer to the stationary distribution, and use relatively fewer samples early on.

To this end, we leverage techniques derived from the MCEM literature in statistics [BH99, LC01, RRT99] to first evaluate a suitable "test" function of the target parameters

at certain subsampled steps, using the gathered samples and current M step estimates. We then use confidence intervals created around these evaluations to gauge the relative effect of successive MCEM estimates over the Monte Carlo error. If the updated values of these functions using newer M-step estimates lie in these intervals, we increase the number of samples collected in the next MCEM loop.

Specifically, similar to [LC01], we start off with the following test function for HMC-EM (Alg. 3): $\mathbf{q}(\cdot) = \left[M^{-1}\mathbf{p}, \nabla\mathscr{L}(\boldsymbol{\theta})\right]$. We then subsample some timesteps as mentioned below, evaluate $\mathbf{q}$ at those steps, and create confidence intervals using sample means and variances:

$$m_S = \frac{1}{S}\sum_{s=1}^{S}\mathbf{q}_s, \quad v_S = \frac{1}{S}\sum_{s=1}^{S}\mathbf{q}_s^2 - m_S^2, \quad C_S := m_S \pm z_{\frac{1-\alpha}{2}} v_S,$$

where S denotes the subsample count, $z_{\frac{1-\alpha}{2}}$ is the $(1-\alpha)$ critical value of a standard Gaussian, and C_S the confidence interval mentioned earlier. For SGNHT-EM (Alg. 4), we use the following test function: $\mathbf{q}(\cdot) = \left[M^{-1}\mathbf{p}, \nabla\mathscr{L}(\boldsymbol{\theta}) + \xi M^{-1}\mathbf{p}, \mathbf{p}^T M^{-1}\mathbf{p}\right]$, derived from the SGNHT dynamics.

One can adopt the following method described in [RRT99]: choose the subsampling offsets $\{t_1 \ldots t_S\}$ as $t_s = \sum_{i=1}^{s} x_i$, where $x_i - 1 \sim \text{Poisson}(\nu i^d)$, with suitably chosen $\nu \geq 1$ and $d > 0$. We found both this and a fixed set of S offsets to work well in our experiments.

With the subsamples collected using this mechanism, we calculate the confidence intervals as described earlier. The assumption is that this interval provides an estimate of the spread of $\mathbf{q}$ due to the Monte Carlo error. We then perform the M-step, and evaluate $\mathbf{q}$ using the updated M-step estimates. If this value lies in the previously calculated confidence bound, we increase S as $S = S + S/S_I$ in the following iteration to overcome the Monte Carlo noise. See [BH99, LC01] for details on these procedures. Values for the constants ν, α, d, S_I, as well as initial estimates for S are given in the supplementary. Running values for S are denoted S_count hereafter.

6.2.3 An Online Update for the M-Step

Algorithm 4 SGNHT-EM

Input: $\boldsymbol{\theta}^{(0)}, \varepsilon, A, LP\_S, S\_\text{count}$
· Initialize $\xi^{(0)}$, $\mathbf{p}^{(0)}$ and M;
repeat
  for $i = 1$ **to** LP_S **do**
    · $\mathbf{p}^{(i+1)} \leftarrow \mathbf{p}^{(i)} - \varepsilon\xi^{(i)}M^{-1}\mathbf{p}^{(i)} - \varepsilon\tilde{\nabla}\mathscr{L}(\boldsymbol{\theta}^{(i)}) + \sqrt{2A}\mathscr{N}(0,\varepsilon)$;
    · $\boldsymbol{\theta}^{(i+1)} \leftarrow \boldsymbol{\theta}^{(i)} + \varepsilon M^{-1}\mathbf{p}^{(i+1)}$;
    · $\xi^{(i+1)} \leftarrow \xi^{(i)} + \varepsilon\left[\frac{1}{D}\mathbf{p}^{(i+1)T}M^{-1}\mathbf{p}^{(i+1)} - 1\right]$;
  end for
  · Set $\left(\boldsymbol{\theta}^{(t+1)}, \mathbf{p}^{(t+1)}, \xi^{(t+1)}\right) = \left(\boldsymbol{\theta}^{(LP\_S+1)}, \mathbf{p}^{(LP\_S+1)}, \xi^{(LP\_S+1)}\right)$;
  · Store MC-EM sample $\mathbf{p}^{(t+1)}$;
  if $(t+1) \bmod S\_\text{count} = 0$ **then**
    · Update M using MC-EM samples;
  end if
  · Update $S_$count as described in the text;
until forever

Next we turn our attention to the task of updating the mass matrices using the collected momenta samples. As shown in the energy functions above, the momenta are sampled from zero-mean normal distributions, enabling us to use standard covariance estimation techniques from the literature. However, since we are using discretized MCMC to obtain these samples, we have to address the variance arising from the Monte Carlo error, especially during the burn-in phase. To that end, we found a running average of the updates to work well in our experiments; in particular, we updated the *inverse* mass matrix, denoted as M_I, at the k^{th} M-step as:

$$M_I^{(k)} = (1 - \kappa^{(k)})M_I^{(k-1)} + \kappa^{(k)}M_I^{(k,\text{est})}, \tag{6.5}$$

where $M_I^{(k,\text{est})}$ is a suitable estimate computed from the gathered samples in the k^{th} M-step, and $\left\{\kappa^{(k)}\right\}$ is a step sequence satisfying some standard assumptions, as described below. Note that the M_Is correspond to the precision matrix of the Gaussian distribution of the momenta; updating this during the M-step also removes the need to invert the mass matrices

during the leapfrog iterations. Curiously, we found the inverse of the empirical covariance matrix to work quite well as $M_I^{(k,\text{est})}$ in our experiments.

These updates also induce a fresh perspective on the convergence of the overall MCEM procedure. Existing convergence analyses in the statistics literature fall into three broad categories: a) the almost sure convergence presented in [FM03] as $t \to \infty$ with increasing sample sizes, b) the asymptotic angle presented in [CL95], where the sequence of MCEM updates are analyzed as an approximation to the standard EM sequence as the sample size, referred to as *S*_count above, tends to infinity, and c) the asymptotic consistency results obtained from multiple Gibbs chains in [SHD99], by letting the chain counts and iterations tend to ∞. Our analysis differs from all of these, by focusing on the maximum likelihood situations noted above as convex optimization problems, and using SGD convergence techniques [Bot98] for the sequence of iterates $M_I^{(k)}$.

Proposition 1. *Assume the* $M_I^{(k,est)}$*'s provide an unbiased estimate of* ∇J*, and have bounded eigenvalues. Let* $\inf_{\|M_I - M_I^*\|^2 > \varepsilon} \nabla J(M_I) > 0 \ \forall \varepsilon > 0$*. Further, let the sequence* $\left\{\kappa^{(k)}\right\}$ *satisfy* $\sum_k \kappa^{(k)} = \infty$*,* $\sum_k \left(\kappa^{(k)}\right)^2 < \infty$*. Then the sequence* $\left\{M_I^{(k)}\right\}$ *converges to the MLE of the precision almost surely.*

Recall that the (negative) *precision* is a natural parameter of the normal distribution written in exponential family notation, and that the log-likelihood is a concave function of the natural parameters for this family; this makes max-likelihood a convex optimization problem over the precision, even in the presence of linear constraints [Uhl12, Dem72]. Therefore, this implies that the problems (6.3), (6.4) have a unique maximum, denoted by M_I^* above. Also note that the update (6.8) corresponds to a first order update on the iterates with an L2-regularized objective, with unit regularization parameter; this is denoted by $J(M_I)$ in the proposition. That is, J is the energy preserved by our sampler(s), as a function of the mass (precision), augmented with an L2 regularization term. The resultant strongly

convex optimization problem can be analyzed using SGD techniques under the assumptions noted above; we provide a proof in the supplementary for completeness.

We should note here that the "stochasticity" in the proof does not refer to the stochastic gradients of $\mathscr{L}(\boldsymbol{\theta})$ used in the leapfrog dynamics of Algorithms 4 through 5; instead we think of the collected momenta samples as a stochastic minibatch used to compute the gradient of the regularized energy, as a function of the covariance (mass), allowing us to deal with the Monte Carlo error indirectly. Also note that our assumption on the unbiasedness of the $M_I^{(k,\text{est})}$ estimates is similar to [FM03], and distinct from assuming that the MCEM samples of $\boldsymbol{\theta}$ are unbiased; indeed, it would be difficult to make this latter claim, since stochastic samplers in general are known to have a convergent bias.

6.2.4 Nosé-Poincaré Variants

We next develop a stochastic version of the dynamics derived from the Nosé-Poincaré Hamiltonian, followed by an MCEM variant. This allows for a direct comparison of the Riemann manifold formulation and our MCEM framework for learning the kinetic masses, in a stochastic setting with thermostat controls on the momentum terms and desired properties like reversibility and symplecticness provided by generalized leapfrog discretizations. The Nosé-Poincaré energy function can be written as [RKP16, BLL99]:

$$H_{NP} = s\left[-\mathscr{L}(\boldsymbol{\theta}) + \frac{1}{2}\left(\frac{\mathbf{p}}{s}\right) M^{-1}\left(\frac{\mathbf{p}}{s}\right) + \frac{q^2}{2Q} + gkT\log s - H_0\right], \tag{6.6}$$

where $\mathscr{L}(\boldsymbol{\theta})$ is the joint log-likelihood, s is the thermostat control, $\mathbf{p}$ and q the momentum terms corresponding to $\boldsymbol{\theta}$ and s respectively, and M and Q the respective mass terms. See [RKP16, BLL99] for descriptions of the other constants. Our goal is to learn both M and Q using the MCEM framework, as opposed to [RKP16], where both were formulated in terms

of $\boldsymbol{\theta}$. To that end, we propose the following system of equations for the stochastic scenario:

$$\begin{aligned}
&\mathbf{p}^{t+\frac{\varepsilon}{2}} = \mathbf{p} + \frac{\varepsilon}{2}\left[s\tilde{\nabla}\mathscr{L}(\boldsymbol{\theta}) - \frac{B(\boldsymbol{\theta})}{\sqrt{s}}M^{-1}\mathbf{p}^{t+\frac{\varepsilon}{2}}\right], \quad \frac{\varepsilon}{4Q}(q^{t+\frac{\varepsilon}{2}})^2 + \left[1 + \frac{A(\boldsymbol{\theta})s\varepsilon}{2Q}\right]q^{t+\frac{\varepsilon}{2}} \\
&\quad - \left[q + \frac{\varepsilon}{2}\left[-gkT(1+\log s) + \frac{1}{2}\left(\frac{\mathbf{p}^{t+\frac{\varepsilon}{2}}}{s}\right)M^{-1}\left(\frac{\mathbf{p}^{t+\frac{\varepsilon}{2}}}{s}\right) + \tilde{\mathscr{L}}(\boldsymbol{\theta}) + H_0\right]\right] = 0, \\
&s^{t+\varepsilon} = s + \varepsilon\left[\frac{q^{t+\frac{\varepsilon}{2}}}{Q}\left(s + s^{t+\frac{\varepsilon}{2}}\right)\right], \quad \boldsymbol{\theta}^{t+\varepsilon} = \boldsymbol{\theta} + \varepsilon M^{-1}\mathbf{p}\left[\frac{1}{s} + \frac{1}{s^{t+\varepsilon}}\right], \\
&\mathbf{p}^{t+\varepsilon} = \mathbf{p}^{t+\frac{\varepsilon}{2}} + \frac{\varepsilon}{2}\left[s^{t+\varepsilon}\tilde{\nabla}\mathscr{L}(\boldsymbol{\theta}^{t+\varepsilon}) - \frac{B(\boldsymbol{\theta}^{t+\varepsilon})}{\sqrt{s^{t+\varepsilon}}}M^{-1}\mathbf{p}^{t+\frac{\varepsilon}{2}}\right], \quad q^{t+\varepsilon} = q^{t+\frac{\varepsilon}{2}} + \frac{\varepsilon}{2}\Bigg[H_0 + \tilde{\mathscr{L}}(\boldsymbol{\theta}^{t+\varepsilon}) \\
&\quad - gkT(1+\log s^{t+\varepsilon}) + \frac{1}{2}\left(\frac{\mathbf{p}^{t+\frac{\varepsilon}{2}}}{s^{t+\varepsilon}}\right)M^{-1}\left(\frac{\mathbf{p}^{t+\frac{\varepsilon}{2}}}{s^{t+\varepsilon}}\right) - \frac{A(\boldsymbol{\theta})s^{t+\varepsilon}}{2Q}q^{t+\frac{\varepsilon}{2}} - \frac{\left(q^{t+\frac{\varepsilon}{2}}\right)^2}{2Q}\Bigg],
\end{aligned} \tag{6.7}$$

where $t+\frac{\varepsilon}{2}$ denotes the half-step dynamics, $\tilde{}$ signifies noisy stochastic estimates, and $A(\boldsymbol{\theta})$ and $B(\boldsymbol{\theta})$ denote the stochastic noise terms, necessary for the Fokker-Planck corrections [RKP16]. Note that we only have to solve a quadratic equation for $q^{t+\frac{\varepsilon}{2}}$ with the other updates also being closed-form, as opposed to the implicit system of equations in [RKP16].

Proposition 2. *The dynamics* (6.7) *preserve the Nosé-Poincaré energy* (6.6).

The proof is a straightforward application of the Fokker-Planck corrections for stochastic noise to the Hamiltonian dynamics derived from (6.6), and is provided in the supplementary. With these dynamics, we first develop the SG-NPHMC algorithm (Alg. 4 in the supplementary) as a counterpart to SGHMC and SGNHT, and wrap it in our MCEM framework to create SG-NPHMC-EM (Alg. 5 in the supplementary). As we shall demonstrate shortly, this EM variant performs comparably to SGR-NPHMC from [RKP16], while being significantly faster.

6.3 Experiments

In this section we compare the performance of the MCEM-augmented variants of HMC, SGHMC as well as SGNHT with their standard counterparts, where the mass matrices are

set to the identity matrix. We call these augmented versions HMC-EM, SGHMC-EM, and SGNHT-EM respectively. As baselines for the synthetic experiments, in addition to the standard samplers mentioned above, we also evaluate RHMC [GC11] and SGR-NPHMC [RKP16], two recent algorithms based on dynamic Riemann manifold formulations for learning the mass matrices. In the topic modeling experiment, for scalability reasons we evaluate only the stochastic algorithms, including the recently proposed SGR-NPHMC, and omit HMC, HMC-EM and RHMC. Since we restrict the discussions in this paper to samplers with second-order dynamics, we do not compare our methods with SGLD [WT11] or SGRLD [PT13].

6.3.1 Parameter Estimation of a 1D Standard Normal Distribution

In this experiment we aim to learn the parameters of a unidimensional standard normal distribution in both batch and stochastic settings, using $5,000$ data points generated from $\mathcal{N}(0,1)$, analyzing the impact of our MC-EM framework on the way. We compare all the algorithms mentioned so far: HMC, HMC-EM, SGHMC, SGHMC-EM, SGNHT, SGNHT-EM, SG-NPHMC, SG-NPHMC-EM along with RHMC and SGR-NPHMC. The generative model consists of normal-Wishart priors on the mean μ and precision τ, with posterior distribution $p(\mu,\tau|\mathbf{X}) \propto N(\mathbf{X}|\mu,\tau)\mathcal{W}(\tau|1,1)$, where $\mathcal{W}$ denotes the Wishart distribution. We run all the algorithms for the same number of iterations, discarding the first $5,000$ as "burn-in". Batch sizes were fixed to 100 for all the stochastic algorithms, along with 10 leapfrog iterations across the board. For SGR-NPHMC and RHMC, we used the observed Fisher information plus the negative Hessian of the prior as the tensor, with one fixed point iteration on the implicit system of equations arising from the dynamics of both. For HMC we used a fairly high learning rate of $1e-2$. For SGHMC and SGNHT we used $A = 10$ and $A = 1$ respectively. For SGR-NPHMC we used $A, B = 0.01$.

We show the RMSE numbers collected from post-burn-in samples as well as per-iteration runtimes in Table 1. An "iteration" here refers to a complete E step, with the full quota of leapfrog jumps. The improvements afforded by our MCEM framework are immediately noticeable; HMC-EM matches the errors obtained from RHMC, in effect matching the sample distribution, while being much faster (an order of magnitude) per iteration. The stochastic MCEM algorithms show markedly better performance as well; SGNHT-EM in particular beats SGR-NPHMC in RMSE-τ while being significantly faster due to simpler updates for the mass matrices. Accuracy improvements are particularly noticeable for the high learning rate regimes for HMC, SGHMC and SG-NPHMC.

METHOD	RMSE (μ)	RMSE (τ)	TIME
HMC	0.0196	0.0197	0.417MS
HMC-EM	0.0115	0.0104	0.423MS
RHMC	0.0111	0.0089	5.748MS
SGHMC	0.1590	0.1646	0.133MS
SGHMC-EM	0.0713	0.2243	0.132MS
SG-NPHMC	0.0326	0.0433	0.514MS
SG-NPHMC-EM	0.0274	0.0354	0.498MS
SGR-NPHMC	0.0240	0.0308	3.145MS
SGNHT	0.0344	0.0335	0.148MS
SGNHT-EM	0.0317	0.0289	0.148MS

Table 6.1: RMSE of the sampled means, precisions and per-iteration runtimes (in milliseconds) from runs on synthetic Gaussian data.

6.3.2 Parameter Estimation in 2D Bayesian Logistic Regression

Next we present some results obtained from a Bayesian logistic regression experiment, using both synthetic and real datasets. For the synthetic case, we used the same methodology as [RKP16]; we generated $2,000$ observations from a mixture of two normal distributions with means at $[1,-1]$ and $[-1,1]$, with mixing weights set to $(0.5,0.5)$ and the covariance set to I. We then classify these points using a linear classifier with weights $\{W_0,W_1\} = [1,-1]$, and attempt to learn these weights using our samplers. We put $\mathcal{N}(0,10I)$ priors on the weights, and used the metric tensor described in §7 of [GC11]

for the Riemannian samplers. In the (generalized) leapfrog steps of the Riemannian samplers, we opted to use 2 or 3 fixed point iterations to approximate the solutions to the implicit equations. Along with this synthetic setup, we also fit a Bayesian LR model to the Australian Credit and Heart regression datasets from the UCI database, for additional runtime comparisons. The Australian credit dataset contains 690 datapoints of dimensionality 14, and the Heart dataset has 270 13-dimensional datapoints.

For the synthetic case, we discard the first $10,000$ samples as burn-in, and calculate RMSE values from the remaining samples. Learning rates were chosen from $\{1e-2, 1e-4, 1e-6\}$, and values of the stochastic noise terms were selected from $\{0.001, 0.01, 0.1, 1, 10\}$. Leapfrog steps were chosen from $\{10, 20, 30\}$. For the stochastic algorithms we used a batchsize of 100.

METHOD	RMSE (W_0)	RMSE (W_1)
HMC	0.0456	0.1290
HMC-EM	0.0145	0.0851
RHMC	0.0091	0.0574
SGHMC	0.2812	0.2717
SGHMC-EM	0.2804	0.2583
SG-NPHMC	0.4945	0.4263
SG-NPHMC-EM	0.0990	0.4229
SGR-NPHMC	0.1901	0.1925
SGNHT	0.2035	0.1921
SGNHT-EM	0.1983	0.1729

Table 6.2: RMSE of the two regression parameters, for the synthetic Bayesian logistic regression experiment. See text for details.

The RMSE numbers for the synthetic dataset are shown in Table 2, and the per-iteration runtimes for all the datasets are shown in Table 3. We used initialized S_count to 300 for HMC-EM, SGHMC-EM, and SGNHT-EM, and 200 for SG-NPHMC-EM. The MCEM framework noticeably improves the accuracy in almost all cases, with no computational overhead. Note the improvement for SG-NPHMC in terms of RMSE for W_0. For the runtime calculations, we set all samplers to 10 leapfrog steps, and fixed S_count to the values mentioned above.

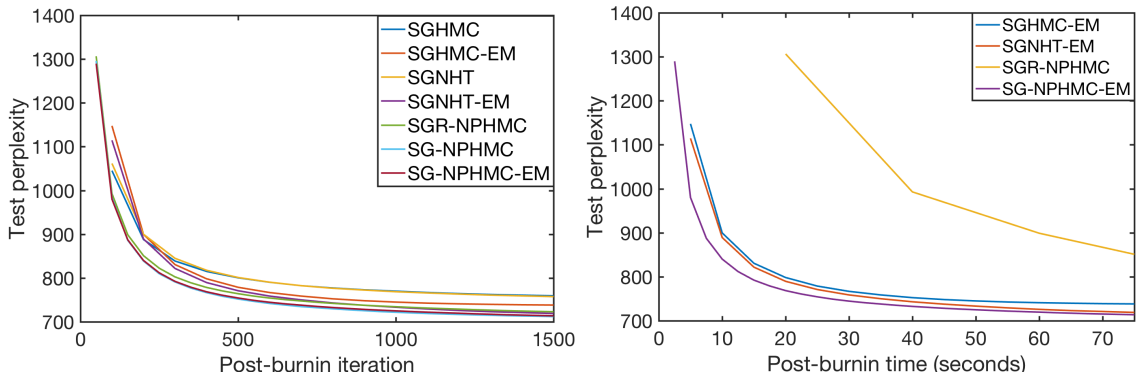
METHOD	TIME (SYNTH)	TIME (AUS)	TIME (HEART)
HMC	1.435MS	0.987MS	0.791MS
HMC-EM	1.428MS	0.970MS	0.799MS
RHMC	1550MS	367MS	209MS
SGHMC	0.200MS	0.136MS	0.112MS
SGHMC-EM	0.203MS	0.141MS	0.131MS
SG-NPHMC	0.731MS	0.512MS	0.403MS
SG-NPHMC-EM	0.803MS	0.525MS	0.426MS
SGR-NPHMC	6.720MS	4.568MS	3.676MS
SGNHT	0.302MS	0.270MS	0.166MS
SGNHT-EM	0.306MS	0.251MS	0.175MS

Table 6.3: Per-iteration runtimes (in milliseconds) for Bayesian logistic regression experiments, on both synthetic and real datasets.

The comparisons with the Riemannian algorithms tell a clear story: though we do get somewhat better accuracy with these samplers, they are orders of magnitude slower. In our synthetic case, for instance, each iteration of RHMC (consisting of all the leapfrog steps and the M-H ratio calculation) takes more than a second, using 10 leapfrog steps and 2 fixed point iterations for the implicit leapfrog equations, whereas both HMC and HMC-EM are simpler and much faster. Also note that the M-step calculations for our MCEM framework involve a single-step closed form update for the precision matrix, using the collected samples of $\mathbf{p}$ once every S_count sampling steps; thus we can amortize the cost of the M-step over the previous S_count iterations, leading to negligible changes to the per-sample runtimes.

6.3.3 Topic Modeling using a Nonparametric Gamma Process Construction

Next we turn our attention to a high-dimensional topic modeling experiment using a nonparametric Gamma process construction. We elect to follow the experimental setup described in [RKP16]. Specifically, we use the Poisson factor analysis framework of [ZC15]. Denoting the vocabulary as V, and the documents in the corpus as D, we model the observed counts of the vocabulary terms as $\mathbf{D}_{V\times N} = \text{Poi}(\mathbf{\Phi\Theta})$, where $\mathbf{\Theta}_{K\times N}$ models the counts of K latent topics in the documents, and $\mathbf{\Phi}_{V\times K}$ denotes the factor load matrix, that encodes the



(a) (b)

Figure 6.1: Test perplexities plotted against (a) post-burnin iterations and (b) wall-clock time for the 20-Newsgroups dataset. See text for experimental details.

relative importance of the vocabulary terms in the latent topics. Following standard Bayesian convention, we put model the columns of $\mathbf{\Phi}$ as $\phi_{\cdot,k} \sim \text{Dirichlet}(\alpha)$, using normalized Gamma variables: $\phi_{v,k} = \frac{\gamma_v}{\sum_v \gamma_v}$, with $\gamma_v \sim \Gamma(\alpha, 1)$. Then we have $\theta_{n,k} \sim \Gamma(r_k, \frac{p_j}{1-p_j})$; we put $\beta(a_0, b_0)$ priors on the document-specific mixing probabilities p_j. We then set the r_ks to the atom weights generated by the constructive Gamma process definition of [RK15]; we refer the reader to that paper for the details of the formulation. It leads to a rich nonparametric construction of this Poisson factor analysis model for which closed-form Gibbs updates are infeasible, thereby providing a testing application area for the stochastic MCMC algorithms. We omit the Metropolis Hastings correction-based HMC and RHMC samplers in this evaluation due to poor scalability.

We use count matrices from the 20-Newsgroups and Reuters Corpus Volume 1 corpora [SSH13]. The former has $2,000$ words and $18,845$ documents, while the second has a vocabulary of size $10,000$ over $804,414$ documents. We used a chronological $60-40$ train-test split for both datasets. Following standard convention for stochastic algorithms, following each minibatch we learn document-specific parameters from 80% of the test set,

and calculate test perplexities on the remaining 20%. Test perplexity, a commonly used measure for such evaluations, is detailed in the supplementary.

As noted in [RK15], the atom weights have three sets of components: the E_ks, T_ks and the hyperparameters α, γ and c. As in [RKP16], we ran three parallel chains for these parameters, collecting samples of the momenta from the T_k and hyperparameter chains for the MCEM mass updates. We kept the mass of the E_k chain fixed to I_K, and chose $K = 100$ as number of latent topics. We initialized S_count, the E-step sample size in our algorithms, to 50 for NPHMC-EM and 100 for the rest. Increasing S_count over time yielded fairly minor improvements, hence we kept it fixed to the values above for simplicity. Additional details on batch sizes, learning rates, stochastic noise estimates, leapfrog iterations etc are provided in the supplementary. For the 20-Newsgroups dataset we ran all algorithms for $1,500$ burn-in iterations, and collected samples for the next $1,500$ steps thereafter, with a stride of 100, for perplexity calculations. For the Reuters dataset we used $2,500$ burn-in iterations. Note that for all these algorithms, an "iteration" corresponds to a full E-step with a stochastic minibatch.

The numbers obtained at the end of the runs are shown in Table 2, along with per-iteration runtimes. The post-burnin perplexity-vs-iteration plots from the 20-Newsgroups dataset are shown in Figure 6.1. We can see significant improvements from the MCEM framework for all samplers, with that of SGNHT

METHOD	20-NEWS	REUTERS	TIME(20-NEWS)
SGHMC	759	996	0.047S
SGHMC-EM	738	972	0.047S
SGNHT	757	979	0.045S
SGNHT-EM	719	968	0.045S
SGR-NPHMC	723	952	0.410S
SG-NPHMC	714	958	0.049S
SG-NPHMC-EM	712	947	0.049S

Table 6.4: Test perplexities and per-iteration runtimes on 20-Newsgroups and Reuters datasets.

being highly pronounced (719 vs 757); indeed, the SG-NPHMC samplers have lower perplexities (712) than those obtained by SGR-NPHMC (723), while being close to an order of magnitude faster per iteration for 20-Newsgroups even when the latter used diagonalized metric tensors, ostensibly by avoiding implicit systems of equations in the leapfrog steps to learn the kinetic masses. The framework yields nontrivial improvements for the Reuters dataset as well.

6.4 Conclusion

We propose a new theoretically grounded approach to learning the mass matrices in Hamiltonian-based samplers, including both standard HMC and stochastic variants, using a Monte Carlo EM framework. In addition to a newly proposed stochastic sampler, we augment certain existing samplers with this technique to devise a set of new algorithms that learn the kinetic masses dynamically from the data in a flexible and scalable fashion. Experiments conducted on synthetic and real datasets demonstrate the efficacy and efficiency of our framework, when compared to existing Riemannian manifold-based samplers.

6.5 Proof Details and Experimental Addenda

6.5.1 Proposition 1: Convergence Discussion

We propose the following update for the precision / inverse mass matrix, denoted as M_I, at the k^{th} M-step:

$$M_I^{(k)} = (1 - \kappa^{(k)})M_I^{(k-1)} + \kappa^{(k)} M_I^{(k,\text{est})}, \tag{6.8}$$

where $M_I^{(k,\text{est})}$ is the estimate computed from the gathered samples in the k^{th} M-step, and $\left\{\kappa^{(k)}\right\}$ is a step sequence satisfying some standard assumptions, as described below.

Proposition 1. *Assume the $M_I^{(k,est)}$'s provide an unbiased estimate of ∇J, and have bounded eigenvalues. Let $\inf_{\|M_I - M_I^*\|^2 > \varepsilon} \nabla J(M_I) > 0 \ \forall \varepsilon > 0$. Further, let the sequence $\left\{\kappa^{(k)}\right\}$ satisfy $\sum_k \kappa^{(k)} = \infty$, $\sum_k \left(\kappa^{(k)}\right)^2 < \infty$. Then the sequence $\left\{M_I^{(k)}\right\}$ converges to the MLE of the precision almost surely.*

Proof. The proof follows the basic outline laid out in [Bot98]. With a slight abuse of notation, we use M_k to denote the iterates, $\bar{M}_k$ to denote $M_I^{(k,\text{est})}$, and replace the $\kappa^{(k)}$s with κ_k. Then, as mentioned in the main text, the update (6.8) can be written in the following first-order form:

$$M_k = M_{k-1} + \kappa_k \nabla J(M_k),$$

where $J(\cdot)$ is the L2-regularized energy mentioned in the main text, as a function of the precision, and we assume $\mathbb{E}_z \bar{M}_k(z) = \nabla J(M_k)$, z being a random variable codifying the stochasticity in the estimate $\bar{M}_k$. As mentioned in the main paper, this stochasticity can be thought of as a surrogate for the Monte Carlo error in the collected momenta samples. Now define the Lyapunov function:

$$h(M_k) = \|M_k - M^*\|^2,$$

where M^* is the unique maximizer of the regularized objective function; as mentioned earlier, this exists because the precision is a natural parameter of the normal written in exponential family form, and the log likelihood of the latter is concave in the natural parameters. Then we can write the difference in Lyapunov errors for successive iterates as

$$h(M_{k+1}) - h(M_k) = -2\kappa_k \left(M_k - M^*\right)^T \bar{M}_k(z_k) + \kappa_k^2 \|\bar{M}_k(z_k)\|^2.$$

Denoting the σ-algebra of all the z variables seen till the k^{th} step by $\mathscr{F}_k$, and using conditional independences of the expectations given this information, we can write the expectation of the quantity above as:

$$\mathbb{E}\left(h(M_{k+1}) - h(M_k)|\,\mathscr{F}_k\right) = -2\kappa_k \left(M_k - M^*\right)^T \nabla J(M_k) + \kappa_k^2 \mathbb{E}\|\bar{M}_k(z_k)\|^2. \tag{6.9}$$

Now, since we assumed the $\bar{M}_k$'s to have bounded eigenvalues, we can bound the expectation on the right above as follows:

$$\mathbb{E}\|\bar{M}_k(z_k)\|^2 \leq A + B\|M_k - M^*\|^2,$$

for sufficiently large values of $A, B \geq 0$. This allows to write 6.9 as follows:

$$\mathbb{E}\left(h(M_{k+1}) - (1-\kappa_k^2 B)h(M_k)|\mathscr{F}_k\right) \leq -2\kappa_k \left(M_k - M^*\right)^T \nabla J(M_k) + \kappa_k^2 A. \tag{6.10}$$

Now we define two sequences as follows:

$$\mu_k = \prod_{i=1}^{k} \frac{1}{1-\kappa_k^2 B}, \quad h'_k = \mu_k h(M_k). \tag{6.11}$$

The sequence $\{\mu_k\}$ can be seen to converge based on our assumptions on κ_k^2. Then we can bound the positive variations of h'_k-s as:

$$\mathbb{E}\left[\mathbb{E}(h'_{k+1} - h'_k)^+\right]|\mathscr{F}_k \leq \kappa_k^2 \mu_k A.$$

This proves h'_k to be a quasi-martingale. By the convergence theorem for quasi-martingales [Fis65], we know that these converge almost surely. Since $\{\mu_k\}$ converge as well, we have

almost sure convergence of the $h(M_k)$'s. Combined with the assumption that $\sum_k \kappa_k = \infty$ and eqn. (6.10), we have almost sure convergence of $(M_k - M^*)^T \nabla J(M_k)$ to 0. The final assumption of the proposition allows us to use this result to prove that $M_k \to M^*$ almost surely.

□

6.5.2 Stochastic samplers with MCEM augmentations

In this section we present the MCEM variant of the SGHMC algorithm [CCG14], followed by the SG-NPHMC algorithm using stochastic dynamics derived from the Nosé-Poincaré Hamiltonian. This is then given the MCEM treatment, leading to the SG-NPHMC-EM method.

6.5.2.1 SGHMC-EM

The MCEM variant of the SGHMC algorithm, which we denote SGHMC-EM, is given in Alg. (3). We simply take the standard HMC dynamics, add Fokker-Planck correction terms to handle the stochastic noise, and use the MCEM framework from the main paper to collect appropriate number of samples of $\mathbf{p}$, and use them to update the mass M. C and $\hat{B}$ are user-specified estimates of the noise in the stochastic gradients.

Algorithm 3 SGHMC-EM

Input: $\boldsymbol{\theta}^{(0)}, \varepsilon, A, LP\_S, S\_\text{count}$
· Initialize $\xi^{(0)}$, $\mathbf{p}^{(0)}$ and M;
repeat
  · Sample $\mathbf{p}^{(t)} \sim N(0, M)$;
  for $i = 1$ **to** LP_S **do**
    · $\mathbf{p}^{(i+1)} \leftarrow \mathbf{p}^{(i)} - \varepsilon C M^{-1}\mathbf{p}^{(i)} - \varepsilon\tilde{\nabla}\mathscr{L}(\boldsymbol{\theta}^{(i)}) + \sqrt{2(C-\hat{B})}\mathscr{N}(0,\varepsilon)$;
    · $\boldsymbol{\theta}^{(i+1)} \leftarrow \boldsymbol{\theta}^{(i)} + \varepsilon M^{-1}\mathbf{p}^{(i+1)}$;
  end for
  · Set $\left(\boldsymbol{\theta}^{(t+1)}, \mathbf{p}^{(t+1)}\right) = \left(\boldsymbol{\theta}^{(LP\_S+1)}, \mathbf{p}^{(LP\_S+1)}\right)$;
  · Store MC-EM sample $\mathbf{p}^{(t+1)}$;
  if $(t+1)$ mod $S\_\text{count} = 0$ **then**
    · Update M using MC-EM samples;
  end if
  · Update $S_$count as described in the text;
until forever

6.5.2.2 SG-NPHMC

As mentioned in the main paper, the Nosé-Poincaré energy function can be written as follows [RKP16, BLL99]:

$$H_{NP} = s\left[-\mathscr{L}(\boldsymbol{\theta}) + \frac{1}{2}\left(\frac{\mathbf{p}}{s}\right)M^{-1}\left(\frac{\mathbf{p}}{s}\right) + \frac{q^2}{2Q} + gkT\log s - H_0\right], \tag{6.12}$$

where $\mathscr{L}(\boldsymbol{\theta})$ is the joint log-likelihood, s is the thermostat control, $\mathbf{p}$ and q the momentum terms corresponding to $\boldsymbol{\theta}$ and s respectively, and M and Q the respective mass terms. See [RKP16, BLL99] for descriptions of the other constants. Our goal is to learn both M and Q using the MCEM framework, as opposed to [RKP16], where both were formulated in terms

of $\boldsymbol{\theta}$. To that end, we propose the following system of equations for the stochastic scenario:

$$\begin{aligned}
&\mathbf{p}^{t+\frac{\varepsilon}{2}} = \mathbf{p} + \frac{\varepsilon}{2}\left[s\tilde{\nabla}\mathscr{L}(\boldsymbol{\theta}) - \frac{B(\boldsymbol{\theta})}{\sqrt{s}}M^{-1}\mathbf{p}^{t+\frac{\varepsilon}{2}}\right], \quad \frac{\varepsilon}{4Q}(q^{t+\frac{\varepsilon}{2}})^2 + \left[1 + \frac{A(\boldsymbol{\theta})s\varepsilon}{2Q}\right]q^{t+\frac{\varepsilon}{2}} \\
&- \left[q + \frac{\varepsilon}{2}\left[-gkT(1+\log s) + \frac{1}{2}\left(\frac{\mathbf{p}^{t+\frac{\varepsilon}{2}}}{s}\right)M^{-1}\left(\frac{\mathbf{p}^{t+\frac{\varepsilon}{2}}}{s}\right) + \tilde{\mathscr{L}}(\boldsymbol{\theta}) + H_0\right]\right] = 0, \\
&s^{t+\varepsilon} = s + \varepsilon\left[\frac{q^{t+\frac{\varepsilon}{2}}}{Q}\left(s + s^{t+\frac{\varepsilon}{2}}\right)\right], \quad \boldsymbol{\theta}^{t+\varepsilon} = \boldsymbol{\theta} + \varepsilon M^{-1}\mathbf{p}\left[\frac{1}{s} + \frac{1}{s^{t+\varepsilon}}\right], \\
&\mathbf{p}^{t+\varepsilon} = \mathbf{p}^{t+\frac{\varepsilon}{2}} + \frac{\varepsilon}{2}\left[s^{t+\varepsilon}\tilde{\nabla}\mathscr{L}(\boldsymbol{\theta}^{t+\varepsilon}) - \frac{B(\boldsymbol{\theta}^{t+\varepsilon})}{\sqrt{s^{t+\varepsilon}}}M^{-1}\mathbf{p}^{t+\frac{\varepsilon}{2}}\right], \quad q^{t+\varepsilon} = q^{t+\frac{\varepsilon}{2}} + \frac{\varepsilon}{2}\Bigg[H_0 \\
&+ \tilde{\mathscr{L}}(\boldsymbol{\theta}^{t+\varepsilon}) - gkT(1+\log s^{t+\varepsilon}) + \frac{1}{2}\left(\frac{\mathbf{p}^{t+\frac{\varepsilon}{2}}}{s^{t+\varepsilon}}\right)M^{-1}\left(\frac{\mathbf{p}^{t+\frac{\varepsilon}{2}}}{s^{t+\varepsilon}}\right) - \frac{A(\boldsymbol{\theta})s^{t+\varepsilon}}{2Q}q^{t+\frac{\varepsilon}{2}} - \frac{\left(q^{t+\frac{\varepsilon}{2}}\right)^2}{2Q}\Bigg]
\end{aligned} \tag{6.13}$$

where $t+\frac{\varepsilon}{2}$ denotes the half-step dynamics, $\tilde{}$ signifies noisy stochastic estimates, and $A(\boldsymbol{\theta})$ and $B(\boldsymbol{\theta})$ denote the stochastic noise terms, necessary for the Fokker-Planck corrections [RKP16].

Proposition 2. *The dynamics* (6.13) *preserve the Nosé-Poincaré energy* (6.12).

Proof. We start off with the basic dynamics derived from the Nosé-Poincaré Hamiltonian:

$$\begin{aligned}
\dot{\boldsymbol{\theta}} &= M^{-1}\frac{\mathbf{p}}{s} \\
\dot{\mathbf{p}} &= s\nabla\mathscr{L}(\boldsymbol{\theta}) \\
\dot{s} &= \frac{q}{Q}s \\
\dot{q} &= \mathscr{L}(\boldsymbol{\theta}) + \frac{1}{2}\left(\frac{\mathbf{p}}{s}\right)^T M^{-1}\left(\frac{\mathbf{p}}{s}\right) - gkT(1+\log s) - \frac{q^2}{2Q} + H_0,
\end{aligned} \tag{6.14}$$

where the dot notation denotes the time derivatives. Following the notation of [YA06], this can be expressed as:

$$\begin{bmatrix}\dot{\boldsymbol{\theta}} \\ \dot{\mathbf{p}} \\ \dot{s} \\ \dot{q}\end{bmatrix} = -\begin{bmatrix}0 & 0 & 0 & -I \\ 0 & 0 & I & 0 \\ 0 & -I & 0 & 0 \\ I & 0 & 0 & 0\end{bmatrix}\begin{bmatrix}\frac{\partial}{s}H_{NP} \\ \frac{\partial}{q}H_{NP} \\ \frac{\partial}{\boldsymbol{\theta}}H_{NP} \\ \frac{\partial}{\mathbf{p}}H_{NP}\end{bmatrix} + \mathbf{N}, \tag{6.15}$$

where $\mathbf{N} = [0, \mathcal{N}(0, 2\sqrt{s}B(\boldsymbol{\theta})), 0, \mathcal{N}(0, 2B(\boldsymbol{\theta}))]$ would be the stochastic noise from the minibatch estimates of $\nabla\mathscr{L}(\boldsymbol{\theta})$ and $\mathscr{L}(\boldsymbol{\theta})$ respectively. Denoting the first matrix on the right by D and the second by ∇H_{NP}, we can see that $\text{tr}\left\{\nabla^T \nabla D y\right\} = 0$ for any $y = y(\boldsymbol{\theta}, \mathbf{p}, s, q)$.

Recall that the joint distribution of interest, $p(\boldsymbol{\theta}, \mathbf{p}, s, q) \propto \exp(-H_{NP})$; thus $\nabla p(\boldsymbol{\theta}, \mathbf{p}, s, q) = -p\nabla H_{NP}$.

Now, for any stochastic differential equation written as $\dot{\boldsymbol{\theta}} = f(\boldsymbol{\theta}) + \mathcal{N}(0, 2Q(\boldsymbol{\theta}))$, the Fokker-Planck equation can be written as:

$$\frac{\partial}{\partial t} p(\boldsymbol{\theta}) = -\frac{\partial}{\partial \boldsymbol{\theta}}[f(\boldsymbol{\theta})p(\boldsymbol{\theta})] + \frac{\partial^2}{\partial \boldsymbol{\theta}^2}[Q(\boldsymbol{\theta})p(\boldsymbol{\theta})],$$

where $p(\boldsymbol{\theta})$ denotes the distribution of $\boldsymbol{\theta}$, and $\frac{\partial^2}{\partial \boldsymbol{\theta}^2} = \sum_{i,j} \frac{\partial}{\partial \theta_i} \frac{\partial}{\partial \theta_j}$. For our Nosé-Poincaré case, the right hand side can be written as:

$$\begin{aligned}
&\text{tr}\left\{\nabla^T X \nabla p(\boldsymbol{\theta}, \mathbf{p}, s, q)\right\} + \text{tr}\nabla^T \left\{p(\boldsymbol{\theta}, \mathbf{p}, s, q) D \nabla H_{NP}\right\} \\
&= \text{tr}\left\{(X + D)\nabla^T \nabla p(\boldsymbol{\theta}, \mathbf{p}, s, q)\right\} + \text{tr}\nabla^T \left\{p(\boldsymbol{\theta}, \mathbf{p}, s, q) D \nabla H_{NP}\right\},
\end{aligned}$$

where the diffusion noise matrix

$$X = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \sqrt{s}B(\boldsymbol{\theta}) \\ 0 & 0 & 0 & 0 \\ 0 & A(\boldsymbol{\theta}) & 0 & 0 \end{bmatrix}.$$

Thus replacing D by $X + D$ in (6.15) would make the RHS zero. This transformation would add correction terms to the dynamics 6.14 to yield the following:

$$\begin{aligned}
\dot{\boldsymbol{\theta}} &= M^{-1}\frac{\mathbf{p}}{s} \\
\dot{\mathbf{p}} &= s\nabla\mathscr{L}(\boldsymbol{\theta}) - \sqrt{s}B(\boldsymbol{\theta})M^{-1}\frac{\mathbf{p}}{s} \\
\dot{s} &= \frac{q}{Q}s \\
\dot{q} &= \mathscr{L}(\boldsymbol{\theta}) + \frac{1}{2}\left(\frac{\mathbf{p}}{s}\right)^T M^{-1}\left(\frac{\mathbf{p}}{s}\right) - gkT(1 + \log s) - \frac{q^2}{2Q} - A(\boldsymbol{\theta})\frac{q}{Q}s + H_0.
\end{aligned}$$

Discretizing this system using the generalized leapfrog technique gives rise to the dynamics 6.7.

□

The dynamics 6.13 therefore induce the SG-NPHMC algorithm, shown in Alg. (4).

Algorithm 4 SG-NPHMC

Input: $\boldsymbol{\theta}^{(0)}, \varepsilon, A, LP\_S, S\_\text{count}$
· Initialize $\mathbf{p}^{(0)}$, M, Q;
repeat
  · Sample $\mathbf{p}^{(t)} \sim N(0, M)$, $q \sim N(0, Q)$;
  for $i = 1$ **to** LP_S **do**
    · Perform generalized leapfrog dynamics (6.13) to get $\mathbf{p}^{(i+\varepsilon)}, \boldsymbol{\theta}^{(i+\varepsilon)}, s^{(i+\varepsilon)}, q^{(i+\varepsilon)}$;
  end for
  · Set $\left(\boldsymbol{\theta}^{(t+1)}, \mathbf{p}^{(t+1)}, \xi^{(t+1)}\right) = \left(\boldsymbol{\theta}^{(LP\_S+\varepsilon)}, \mathbf{p}^{(LP\_S+\varepsilon)}, s^{(LP\_S+\varepsilon)}, q^{(LP\_S+\varepsilon)}\right)$;
until forever

6.5.2.3 SG-NPHMC-EM

In this section we add the MCEM framework to Alg. (4) above. This allows us to learn M adaptively while preserving the thermostat controls and symplecticness of the SG-NPHMC sampler.

Algorithm 5 SG-NPHMC-EM

Input: $\boldsymbol{\theta}^{(0)}, \varepsilon, A, B, LP\_S, S\_\text{count}$
· Initialize $s^{(0)}$, $\mathbf{p}^{(0)}$, $q^{(0)}$, M and Q;
repeat
  for $i = 1$ **to** LP_S **do**
    · Perform generalized leapfrog dynamics (6.13) to get $\mathbf{p}^{(i+\varepsilon)}, \boldsymbol{\theta}^{(i+\varepsilon)}, s^{(i+\varepsilon)}, q^{(i+\varepsilon)}$;
  end for
  · Set $\left(\boldsymbol{\theta}^{(t+1)}, \mathbf{p}^{(t+1)}, s^{(t+1)}, q^{(t+1)}\right) = \left(\boldsymbol{\theta}^{(LP\_S+\varepsilon)}, \mathbf{p}^{(LP\_S+\varepsilon)}, s^{(LP\_S+\varepsilon)}, q^{(LP\_S+\varepsilon)}\right)$;
  · Store MC-EM samples $\mathbf{p}^{(t+1)}$ and $q^{(t+1)}$;
  if $(t+1)$ mod $S\_\text{count} = 0$ **then**
    · Update M, Q using MC-EM samples of $\mathbf{p}$ and q respectively;
  end if
  · Update $S_$count as described in the text;
until forever

6.5.3 Additional Experimental Details

For the topic modeling case, we used the following perplexity measure, as defined as [ZC15]:

$$\text{Perplexity} = \exp\left(-\frac{1}{Y}\sum_{n=1}^{N_{\text{test}}}\sum_{v=1}^{V} y_{nv}\log m_{nv}\right),$$

where y_{nv} refers to the count of vocabulary item v in held-out test document n, $Y = \sum_{n=1}^{N_{\text{test}}}\sum_{v=1}^{V} y_{nv}$, and $m_{nv} = \sum_{s=1}^{S}\sum_{k=1}^{K}\phi_{vk}^{(s)}\theta_{kn}^{(s)} / \sum_{v=1}^{V}\sum_{s=1}^{S}\sum_{k=1}^{K}\phi_{vk}^{(s)}\theta_{kn}^{(s)}$, where we collect S samples of θ, ϕ, and have K latent topics. For the 20-Newsgroups dataset, we used learning tates of $1e-7$ for the T_k chain, $1e-6$ for the hyperparameter chain, for all the samplers. Stochastic noise estimates were of the order of $1e-2$ for SGHMC, SGNHT and their EM variants, and of the order of $1e-1$ for SG-NPHMC and its EM version. We used minibatches of size 100, and 10 leapfrog iterations for all algorithms. The document-level θ, ϕ were learnt using 20 leapfrog iterations of RHMC [GC11], which we found to mix slightly better than Gibbs.

For the sample size updates, we used $\nu = 1$, $\alpha = 1$, $d = 2$, $S_I = 10$. We initialized $S_$count to 50 for the topic modeling experiments with SG-NPHMC-EM, 100 for all other cases. All experiments were run on a Macbook pro with a 2.5Ghz core i7 processor and 16GB ram.

Chapter 7: Conclusion and Future Work

We conclude with a brief summary of the various algorithms discussed above, as well as the avenues for future work we consider to be promising in this context.

7.1 Contributions.

We begin by restating our thesis statement: *We posit that scalable learning and inference algorithms are necessary for expanding the scope of Bayesian nonparametric probabilistic models to truly large-scale datasets. Doing so requires developing scalable variants of analytic closed form learning methods that preserve many of the theoretical niceties, from both algorithmic and underlying modeling perspectives. To that end, we propose a set of methods using techniques such as stochastic MCMC sampling, stochastic optimization in both Euclidean and Riemannian manifold domains, variational inference and small-variance asymptotics, that can be applied to large scale data modeled using rich Bayesian nonparametric paradigms, while scaling to large datasets.*

In the preceding chapters, we have discussed the following approaches to addressing the problem of robust and scalable inference in Bayesian nonparametric machine learning:

- We have proposed a novel Hamiltonian Monte Carlo technique consisting of a Riemann preconditioning approach for adaptive learning of the mass matrices of the momenta, applied to Nosé-Poincaré Hamiltonians. We have also converted the resultant dynamics into a form suitable for the stochastic setting using Fokker-Planck

updates to account for random noise. Discretized samplers have been derived for both deterministic and stochastic cases, with robust performance shown in both synthetic and high-dimensional real-world datasets.

- We have proposed a novel construction of the infinite Gamma process without using Dirichlet process denormalization, and derived a variational inference algorithm that has been shown to be more scalable than competing Beta-Bernoulli-based approaches for large-scale topic modeling scenarios. This technique can be improved by augmenting it with recent developments in the approximate inference community, as well as adapted to online and asynchronous settings.

- We have discussed an asymptotic derivation of non-probabilistic inference algorithms for Bayesian hidden Markov models, in both parametric and nonparametric cases. This work extended the use of small-variance asymptotic techniques to sequential nonparametric models, and in the process derived a novel algorithm with dynamic programming-based ideas for infinite hidden Markov models, removing the need for dependence on sampling-based methods for dealing with such problems.

- We have proposed the use of variance reduction techniques to devise fast quasi-Newton optimization methods on Riemannian manifolds, as an alternative to constrained optimization in Euclidean space that is the de-facto choice in most machine learning problems. Our stochastic Riemannian L-BFGS algorithm with variance reduction techniques performs strongly in certain common machine learning tasks, compared to state-of-the-art Euclidean methods and first-order Riemannian optimization algorithms, thereby providing important additions to the machine learning practitioner's toolkit.

- We have provided a completely new approach to developing adaptive Monte Carlo sampling algorithms for large scale machine learning applications, as an alternative to recently proposed Riemannian preconditioning methods for implicitly learning the mass matrices in such samplers. The resulting framework can be applied to most existing Hamiltonian samplers in the machine learning literature, stochastic or otherwise, yielding uniform sampling improvements without compromising runtime complexity.

7.2 Future Work.

To motivate this section, we first present some of the limitations of the work in this thesis:

- We proposed a batch variational inference algorithm for our gamma process construction in Chapter 3, which is does not scale particularly well for large high dimensional datasets. One would ideally use stochastic variational inference (SVI) techniques [HBWP13, HB11] to address this issue. Applications of such methods to Bayesian nonparametric models based on completely random measures are in a nascent state of exploration; thus an SVI algorithm for the model we have developed, in addition to being necessary for effective use of our model for very large datasets, would be a useful addition to the ML practitioner's toolkit.

- In the stochastic MCMC sampler based on Nosé-Poincaré dynamics that we discussed in Chapter 4, we did not provide a robust model for the stochastic noise that these samplers generate. There are two noise terms in the dynamics of the sampler, that we denoted by $A(\boldsymbol{\theta})$ and $B(\boldsymbol{\theta})$, that we set to constant scalars in our experiments. Such a naive noise model imposes clear limitations on the robustness of the sampler; indeed, selecting appropriate values for these turned out to be nontrivial part of the

experimental evaluation. Addressing this limitation is likely to improve the stochastic performance of the sampler, in addition to being an interesting research area in its own right.

- As we mentioned in Chapter 5, exponential maps are but one way to approximate the more general concept of a retraction on Riemannian manifolds; there could be many smooth mappings that satisfy the homeomorphic formulation of a retraction. In this work we restrict ourselves to exponential maps due to their straightforward connections with geodesics, whereas ideally one would want a more detailed treatment of such manifold optimization methods for a range of retraction implementations. Another limitation of this work would be our convergence discussion for nonconvex functions on the manifold, where we assumed the eigenvalues of the Hessian approximations generated by our algorithm to be bounded away from zero in some local neighborhood of a local optimum. While this simplifying assumption allowed us to use existing proof techniques from the literature, it sidesteps the issues the multitude of Hessian-based issues that arise in nonconvex scenarios, for example saddle points in Euclidean domains. As mentioned in the Chapter, one usually deals with this in the Euclidean setting by adding some positive scalar to ensure that the eigenvalues remain bounded away from zero, however such "damping" techniques have not been studied properly in the Riemannian setting.

- For the Monte Carlo EM-based framework we proposed in Chapter 6 for learning the mass matrices in HMC-based samplers, we used a direct closed-form expression for covariance estimation in the M-step of our samplers. While the experimental results suggest this to be a reasonable approach for low to moderate dimensional problems, we did not use any high dimensional covariance estimation tricks in our

implementations, leaving open the possibility that the samplers will not perform to their full potential in very high dimensional problems. Note that the work in this Chapter proposes a new framework for learning the mass matrices using Monte Carlo EM; we do not impose any restrictions on the kind of optimization algorithms one would want to use with the gathered momentum samples in the M step. Thus one would ideally want to use robust techniques in these M steps, possibly leveraging the sparsity structure of the intermediate mass matrices to efficiently handle high dimensional parameter spaces.

This leads us to a discussion on certain straightforward ways one can extend and improve upon the work in this thesis; the first of the following sections addresses one of the limitations mentioned above, while the rest discuss ideas that are directly related to the work in this thesis and could potentially be useful additions to the machine learning literature.

7.2.1 Robust modeling of Stochastic Noise Terms in MCMC samplers.

Recall the sampling algorithm from Chapter 4, where we proposed both batch and stochastic versions of a Riemannian Monte Carlo sampler using dynamics derived from a modified Nosé-Poincaré Hamiltonian. As mentioned in section 4.2.2, we introduce noise terms $A(\boldsymbol{\theta})$ and $B(\boldsymbol{\theta})$ to model the stochastic noise arising from the minibatch estimates of the likelihood and the log-likelihood in the discretized dynamics. However, in the experiments we set these to constant values for simplicity. This imposes an independence assumption on the stochastic noise, in addition to introducing the problem of selecting appropriate values for these terms. One would ideally use a more rigorous approach to modeling the stochastic noise; a zero-mean Gaussian model on the noise would be more reasonable, as well as being in line with other areas of research where such "white noise" assumptions are common. One could then use the expected Fisher information as the

covariance, making the term sensitive to the parameter values as well. Such approaches have seen some use in recent literature on stochastic sampling in the ML context [AKW12, SZLS15]: in the former the authors propose a modifications of the stochastic gradient Langevin dynamics algorithm of [WT11] to improve its mixing rate, constructing a Markov chain that samples from a normal surrogate to the true posterior, with covariance given by the empirical Fisher information derived from the stochastic gradients; they appeal to the Bernstein-von-Mises theorem [Cam86] to prove asymptotic correctness. In [SZLS15] the authors used the empirical Fisher information of the gradients to update the Fokker-Planck noise correction term in stochastic gradient Nosé-Hoover dynamics [DFB$^+$14], and showed improved sampling efficiency and mixing rates for discriminative Boltzmann machines [LB08], among other applications. Such methods could be applied to our case as well, and we expect to see improved robustness to stochastic noise as a result.

7.2.2 Coresets for Faster Metropolis-Hastings Corrections.

Another way to improve scalability with reasonable correctness guarantees would be to develop faster ways to perform Metropolis-Hastings correction steps, instead of simply omitting them from the pipeline as is currently done in the stochastic samplers; recall that doing so introduces biases in the convergent distributions. Prior work along these line would include [KCW14] and [BDH14]. where the authors perform sequential hypothesis tests to determine the exact subsample of datapoints to be used for calculating the M-H ratio. We believe that coresets could potentially be used for this purpose. A technique popularized in the computational geometry [AHPV05] and CS theory [HPM04] communities, coresets are beginning to see increased usage in the machine learning literature for designing improvements to clustering algorithms, both for parametric cases like k-means [FMS07, FL11] as well as objective functions like DP-Means derived from Dirichlet process mixture of

Gaussians [BLK15], which happens to be a popular Bayesian nonparametric construct. One way to construct coresets in a clustering context is by using importance sampling on a distribution computed using an approximation to the optimal solution [FL11, BLK15, LS10]; the approximation is computed using problem-specific techniques, and the importance sampling weights are proportional to the *sensitivities* of the datapoints. The sensitivity in this case is a bound on the relative contribution of the datapoint to the cost vs the average contribution of all points. In addition to the importance sampling estimator being unbiased, this scheme can be used to bound the variance of the estimator as well [LS10]. In the sampling scenario, one could potentially develop sequential coreset construction methods aimed at approximating the M-H acceptance ratios upto some specified error, without processing the entire dataset; the biases arising from removing the M-H step altogether could therefore be handled in a methodical way, with suitable complexity trade-offs.

7.2.3 Mixing Rate Analyses.

One could also conduct a thorough analysis of the mixing rates of these stochastic MCMC samplers, along the lines of [LPW08]. There has been some recent work looking at mixing rates of Gibbs samplers for discrete distributions [SOR16], where the authors analyze both mixing time and bias for asynchronous (Hogwild!) Gibbs samplers. Similar rigorous analyses for MCMC samplers for distributions with continuous support would be an interesting contribution to the literature, and would provide sound convergence bounds for these samplers, as opposed to heuristically obtained "burn-in" times.

7.2.4 Parallelization Frameworks for HMC-based samplers.

Lastly, one could investigate ways to parallelize the samplers proposed here; doing so for the Riemannian sampler, for instance, would be nontrivial due to the coordinate dependencies induced by certain formulations of the metric tensor in terms of the parameters.

A robust parallelization framework, with possibly an asynchronous version of these samplers and accompanying theoretical bounds, combined with the techniques for stochastic domains developed here and elsewhere would go a long way towards improving the practical applicability of such algorithms. Insights gained here may also enable practitioners to efficiently run these algorithms on dedicated computer hardware like graphics processing units, which would usher in a new era of rich probabilistic models and algorithms.

To conclude, we hope that the techniques discussed herein will contribute useful techniques to the arsenal of machine learning tools aimed at the problem of scalable parameter estimation and inference in Bayesian nonparametric settings. Given the deluge of data in machine learning contexts, often from increasingly heterogeneous sources, we expect to see an increase in required model complexity to find suitable representations in cases where simple parametric constructions or decades-old heuristics may become a limiting factor in acquiring useful insights. Theoretically grounded probabilistic models such as the ones discussed in this dissertation are likely to see increased usage and adoption in the near future, in which case the algorithms and techniques we develop herein will prove integral to the discerning machine learning researcher's quest to solve scalable inference problems.

Bibliography

[AHPV05] P. K. Agarwal, S. Har-Peled, and K. R. Varadarajan. Geometric Approximation via Coresets. *Combinatorial and Computational Geometry*, 52:1–30, 2005.

[AKW12] S. Ahn, A. Korattikara, and M. Welling. Bayesian Posterior Sampling via Stochastic Gradient Fisher Scoring. In *ICML*, 2012.

[AMS08] P. A. Absil, R. Mahony, and R. Sepulchre. *Optimization Algorithms on Matrix Manifolds*. Princeton University Press, 2008.

[Ant74] C. E. Antoniak. Mixtures of Dirichlet processes with applications to Bayesian nonparametric problems. *The Annals of Statistics*, 2(6):1152–1174, 1974.

[BDH14] R. Bardenet, A. Doucet, and C. Holmes. Towards Scaling up Markov chain Monte Carlo: An Adaptive Subsampling Approach. In *ICML*, 2014.

[BGR02] M. J. Beal, Z. Ghahramani, and C. E. Rasmussen. The infinite hidden Markov model. In *NIPS*, 2002.

[BH99] J. G. Booth and J. P. Hobert. Maximizing Generalized Linear Mixed Model Likelihoods with an Automated Monte Carlo EM Algorithm. *Journal of the Royal Statistical Society Series B*, 61(1):265–285, 1999.

[Bha07] R. Bhatia. *Positive Definite Matrices*. Princeton University Press, 2007.

[BHNS14] R. H. Byrd, S. L. Hansen, J. Nocedal, and Y. Singer. A Stochastic Quasi-Newton Method for Large-scale Optimization, 2014. arXiv:1410.1068.

[BI13] D. A. Bini and B. Iannazzo. Computing the Karcher mean of symmetric positive definite matrices. *Linear Algebra and its Applications*, 483(4):1700–1710, 2013.

[Bis06] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[BJ03] D. Blei and M. Jordan. Variational methods for Dirichlet process mixtures. *Bayesian Analysis*, 1:121–144, 2003.

[BKJ13] T. Broderick, B. Kulis, and M. I. Jordan. MAD-Bayes: MAP-based asymptotic derivations from Bayes. In *ICML*, 2013.

[BL04] L. Bottou and Y. LeCun. Large scale online learning. In *NIPS*, 2004.

[BLK15] O. Bachem, M. Lucic, and A. Krause. Coresets for Nonparametric Estimation – the Case of DP-Means. In *ICML*, 2015.

[BLL99] S. D. Bond, B. J. Leimkuhler, and B. B. Laird. The Nosé-Poincaré Method for Constant Temperature Molecular Dynamics. *J. Comput. Phys*, 151:114–134, 1999.

[BMAS14] N. Boumal, B. Mishra, P.-A. Absil, and R. Sepulchre. Manopt, a Matlab toolbox for optimization on manifolds. *Journal of Machine Learning Research*, 15:1455–1459, 2014.

[BMDG05] A. Banerjee, S. Merugu, I. S. Dhillon, and J. Ghosh. Clustering with Bregman divergences. *Journal of Machine Learning Research*, 6:1705–1749, 2005.

[BMPJ14] T. Broderick, L. Mackey, J. Paisley, and M. I. Jordan. Combinatorial clustering and the beta negative binomial process. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2014.

[Bon13] S. Bonnabel. Stochastic gradient descent on Riemannian manifolds. *IEEE Transactions on Automatic Control*, 58(9):2217–2229, 2013.

[Boo86] W. M. Boothby. *An Introduction to Differentiable Manifolds and Riemannian Geometry*. Academic Press Inc., 1986.

[Bot98] L. Bottou. On-line Learning and Stochastic Approximations. In D. Saad, editor, *On-line Learning in Neural Networks*, pages 9–42. Cambridge University Press, 1998.

[Bot10] L. Bottou. Large-scale machine learning with stochastic gradient descent. In *International Conference on Computational Statistics*, 2010.

[Cam86] L. Le Cam. *Asymptotic Methods in Statistical Decision Theory*. Springer, 1986.

[CCG14] T. Chen, E. Chen, and C. Guestrin. Stochastic Gradient Hamiltonian Monte Carlo. In *ICML*, 2014.

[CF13] F. Caron and E. B. Fox. Bayesian Nonparametric Models of Sparse and Exchangeable Random Graphs, 2013. arXiv:1401.1137.

[CL95] K. S. Chan and J. Ledolter. Monte Carlo EM Estimation for Time Series Models Involving Counts. *Journal of the American Statistical Association*, 90(429):242–252, 1995.

[CTM13] F. Caron, Y. W. Teh, and B. T. Murphy. Bayesian Nonparametric Plackett-Luce models for the Analysis of Clustered Ranked Data, 2013. arXiv:1211.5037.

[Dem72] A. P. Dempster. Covariance selection. *Biometrics*, 28:157–175, 1972.

[DFB+14] N. Ding, Y. Fang, R. Babbush, C. Chen, R. D. Skeel, and H. Neven. Bayesian Sampling using Stochastic Gradient Thermostats. In *NIPS*, 2014.

[Dix82] J. D. Dixon. Exact solution of linear equations using P-adic expansions. *Numerische Mathematik*, 40(1):137–141, 1982.

[DKPR87] S. Duane, A.D. Kennedy, B.J. Pendleton, and D. Roweth. Hybrid Monte Carlo. *Physics Letters B*, 195(2):216–222, 1987.

[DLR77] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society Series B*, 39(1):1–38, 1977.

[DVMGT09] F. Doshi-Velez, K. Miller, J. Van Gael, and Y. W. Teh. Variational Inference for the Indian Buffet Process. In *AISTATS*, 2009.

[EGG+06] W. Eberly, M. Giesbrecht, P. Giorgi, A. Storjohann, and G. Villard. Solving sparse rational linear systems. In *Proceedings of the 2006 international symposium on Symbolic and algebraic computation (ISSAC)*, pages 63–70, 2006.

[Fer73] T. S. Ferguson. A Bayesian Analysis of Some Nonparametric Problems. *The Annals of Statistics*, 1(2):209–230, 1973.

[Fis65] D. L. Fisk. Quasi-Martingales. *Transactions of the American Mathematical Society*, 120(3):369–389, 1965.

[FL11] D. Feldman and M. Langberg. A Unified Framework for Approximating and Clustering Data. In *STOC*, 2011.

[FM03] G. Fort and E. Moulines. Convergence of the Monte Carlo Expectation Maximization for Curved Exponential Families. *The Annals of Statistics*, 31(4):1220–1259, 2003.

[FMS07] D. Feldman, M. Monemizadeh, and C. Sohler. A PTAS for k-Means Clustering Based on Weak Coresets. In *SoCG*, 2007.

[FS01] D. Frenkel and B. Smit. *Understanding Molecular Simulations: From Algorithms to Applications, 2nd Edition*. Academic Press, 2001.

[GC11] M. Girolami and B. Calderhead. Riemann manifold Langevin and Hamiltonian Monte Carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(2):123–214, 2011.

[GCH+15] Z. Gan, C. Chen, R. Henao, D. Carlson, and L. Carin. Scalable Deep Poisson Factor Analysis for Topic Modeling. In *ICML*, 2015.

[GRRB14] P. Gopalan, F. J. R. Ruiz, R. Ranganath, and D.M. Blei. Bayesian nonparametric Poisson factorization. In *AISTATS*, 2014.

[GSTG08] J. V. Gael, Y. Saatci, Y. W. Teh, and Z. Ghahramani. Beam sampling for the infinite hidden Markov model. In *ICML*, 2008.

[HB11] M. D. Hoffman and D. M. Blei. Structured Stochastic Variational Inference. In *AISTATS*, 2011.

[HBWP13] M. D. Hoffman, D. M. Blei, C. Wang, and J. Paisley. Stochastic variational inference. *Journal of Machine Learning Research*, 14:1303–1347, 2013.

[HM15] R. Hosseini and M. Mash'al. Mixest: An Estimation Toolbox for Mixture Models, 2015. arXiv:1507.06065.

[Hoo85] W. G. Hoover. Canonical dynamics: Equilibrium phase-space distributions. *Physical Review A (General Physics)*, 31(3):1695–1697, 1985.

[HPM04] S. Har-Peled and S. Mazumdar. On coresets for k-means and k-median clustering. In *STOC*, 2004.

[HS15] R. Hosseini and S. Sra. Matrix Manifold Optimization for Gaussian Mixtures. In *NIPS*, 2015.

[IJ00] H. Ishwaran and L. F. James. Approximate Dirichlet Process Computing in Finite Normal Mixtures: Smoothing and Prior Information. *Journal of Computational and Graphical Statistics*, 11:508–532, 2000.

[IJ01] H. Ishwaran and L. F. James. Gibbs Sampling Methods for Stick-Breaking Priors. *Journal of the American Statistical Association*, 96:161–173, 2001.

[JKJ12] K. Jiang, B. Kulis, and M. I. Jordan. Small-variance asymptotics for exponential family Dirichlet process mixture models. In *NIPS*, 2012.

[JL11] A. Jones and B. Leimkuhler. Adaptive stochastic methods for sampling driven molecular systems. *Journal of Chemical Physics*, 135(8):084125, 2011.

[Jor10] M. I. Jordan. Hierarchical Models, Nested Models and Completely Random Measures. In M.-H. Chen, D. Dey, P. Mueller, D. Sun, and K. Ye, editors,

Frontiers of Statistical Decision Making and Bayesian Analysis: In Honor of James O. Berger. New York: Springer, 2010.

[JZ13] R. Johnson and T. Zhang. Accelerating Stochastic Gradient Descent using Predictive Variance Reduction. In *NIPS*, 2013.

[KCW14] A. Korattikara, Y. Chen, and M. Welling. Austerity in MCMC Land: Cutting the Metropolis-Hastings Budget. In *ICML*, 2014.

[Kin67] J.F.C. Kingman. Completely Random Measures. *Pacific Journal of Mathematics*, 21(1):59–78, 1967.

[Kin93] J. F. C. Kingman. *Poisson Processes*, volume 3 of *Oxford Studies in Probability*. Oxford University Press, New York, 1993.

[KJ12] B. Kulis and M. I. Jordan. Revisiting k-means: New algorithms via Bayesian nonparametrics. In *ICML*, 2012.

[LB08] H. Larochelle and Y. Bengio. Classification using Discriminative Restricted Boltzmann Machines. In *ICML*, 2008.

[LC01] R. A. Levine and G. Casella. Implementations of the Monte Carlo EM Algorithm. *Journal Computational and Graphical Statistics*, 10(3):422–439, 2001.

[Lee97] J. Lee. *Riemann Manifolds: an Introduction to Curvature*. Springer-Verlag, 1997.

[LM15] B. Leimkuhler and C. Matthews. *Molecular Dynamics: With Deterministic and Stochastic Numerical Methods*. Springer, 2015.

[LN89] D. C. Liu and J. Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45(1–3):503–528, 1989.

[LPW08] D. A. Levin, Y. Peres, and E. L. Wilmer. *Markov Chains and Mixing Times*. AMS, 2008.

[LR05] B. Leimkuhler and S. Reich. *Simulating Hamiltonian Dynamics*. Cambridge University Press, 2005.

[LS10] M. Langberg and L. J. Schulman. Universal ε-approximators for integrals. In *SODA*, 2010.

[McC97] C. E. McCulloch. Maximum Likelihood Algorithms for Generalized Linear Mixed Models. *Journal of the American Statistical Association*, 92(437):162–170, 1997.

[MCF15] Y. Ma, T. Chen, and E. B. Fox. A Complete Recipe for Stochastic Gradient MCMC. In *Advances in Neural Information Processing Systems (NIPS) 28*, pages 2917–2925, 2015.

[Mil11] K. T. Miller. *Bayesian Nonparametric Latent Feature Models*. PhD thesis, University of California at Berkeley, 2011.

[MNJ16] P. Moritz, R. Nishihara, and M. I. Jordan. A Linearly-Convergent Stochastic L-BFGS Algorithm. In *AISTATS*, 2016.

[Nea10] R. M. Neal. Mcmc using Hamiltonian dynamics. In S. Brooks, A. Gelman, G. L. Jones, and X.-L. Meng, editors, *Handbook of Markov Chain Monte Carlo*, pages 113–162. Chapman & Hall / CRC Press, 2010.

[Nos84] S. Nosé. A molecular dynamics method for simulations in the canonical ensemble. *Molecular Physics*, 52(2):255–268, 1984.

[NW06] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, 2006.

[Oja92] E. Oja. Principal components, minor components, and linear neural networks. *Neural Networks*, 5(6):927–935, 1992.

[OW12] P. Orbanz and S. Williamson. Unit-rate Poisson representations of completely random measures, 2012.

[PBJ12] J. Paisley, D. M. Blei, and M. I. Jordan. Stick-Breaking Beta Processes and the Poisson Process. In *Artificial Intelligence and Statistics*, 2012.

[PCB11] J. Paisley, L. Carin, and D. M. Blei. Variational Inference for Stick-Breaking Beta Process Priors. In *ICML*, 2011.

[PT13] S. Patterson and Y. W. Teh. Stochastic Gradient Riemannian Langevin Dynamics on the Probability Simplex. In *NIPS*, 2013.

[PZW+10] J. Paisley, A. Zaas, C. W. Woods, G. S. Ginsburg, and L. Carin. A Stick-Breaking Construction of the Beta Process. In *ICML*, 2010.

[Rab89] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[RF96] J. Ruanaidh and W. J. Fitzgerald. *Numerical Bayesian methods applied to signal processing*. Springer-Verlag New York Inc, 1996.

[RHS+16] S. J. Reddi, A. Hefny, S. Sra, B. Póczós, and A. Smola. Stochastic Variance Reduction for Nonconvex Optimization. In *ICML*, 2016.

[RJK13] A. Roychowdhury, K. Jiang, and B. Kulis. Small Variance Asymptotics for Hidden Markov Models. In *NIPS*, 2013.

[RK15] A. Roychowdhury and B. Kulis. Gamma Processes, Stick-Breaking, and Variational Inference. In *AISTATS*, 2015.

[RKP16] A. Roychowdhury, B. Kulis, and S. Parthasarathy. Robust Monte Carlo Sampling using Riemannian Nosé-Poincaré Hamiltonian Dynamics. In *Proceedings of The 33rd International Conference on Machine Learning (ICML)*, pages 2673–2681, 2016.

[RM51] H. Robbins and S. Monro. A Stochastic Approximation Method. *The Annals of Mathematical Statistics*, 22(3):400–407, 1951.

[Row98] S. Roweis. EM algorithms for PCA and SPCA. In *NIPS*, 1998.

[RP17a] A. Roychowdhury and S. Parthasarathy. Accelerated Stochastic Quasi-Newton Optimization on Riemann Manifolds, 2017. In submission.

[RP17b] A. Roychowdhury and S. Parthasarathy. Adaptive Bayesian Sampling with Monte Carlo EM. In *NIPS*, 2017.

[RRT99] C. P. Robert, T. Rydén, and D. M. Titterington. Convergence Controls for MCMC Algorithms, With Applications to Hidden Markov Chains. *Journal of Statistical Computation and Simulation*, 64:327–355, 1999.

[RW12] W. Ring and B. Wirth. Optimization methods on Riemannian manifolds and their application to shape space. *SIAM Journal on Optimization*, 22(2):596–627, 2012.

[Set94] J. Sethuraman. A constructive definition of Dirichlet priors. *Statistica Sinica*, 4:639–650, 1994.

[SH15] S. Sra and R. Hosseini. Conic geometric optimization on the manifold of positive definite matrices. *SIAM Journal on Optimization*, 25(1):713–739, 2015.

[Sha15] O. Shamir. A Stochastic PCA and SVD Algorithm with an Exponential Convergence Rate. In *ICML*, 2015.

[SHD99] R. P. Sherman, Y.-Y. K. Ho, and S. R. Dalal. Conditions for convergence of Monte Carlo EM sequences with an application to product diffusion modeling . *The Econometrics Journal*, 2(2):248–267, 1999.

[SN09] B. Shahbaba and R. Neal. Nonlinear Models Using Dirichlet Process Mixtures. *JMLR*, 10:1829–1850, 2009.

[SOR16] C. De Sa, K. Olukotun, and C. Ré. Ensuring rapid mixing and low bias for asynchronous Gibbs sampling. In *ICML*, 2016.

[SSH13] N. Srivastava, R. Salakhutdinov, and G. E. Hinton. Modeling documents with deep Boltzmann machines. In *UAI*, 2013.

[Sud12] E. Sudderth. Toward reliable Bayesian nonparametric learning. In *NIPS Workshop on Bayesian Noparametric Models for Reliable Planning and Decision-Making Under Uncertainty*, 2012.

[SZLS15] X. Shang, Z. Zhu, B. Leimkuhler, and A. J. Storkey. Covariance-Controlled Adaptive Langevin Thermostat for Large-Scale Bayesian Sampling. In *NIPS*, 2015.

[TB99] M. E. Tipping and C. M. Bishop. Probabilistic principal component analysis. *Journal of Royal Statistical Society, Series B*, 21(3):611–622, 1999.

[TGG07] Y. W. Teh, D. Görür, and Z. Ghahramani. Stick-breaking construction for the Indian buffet process. In *AISTATS*, volume 11, 2007.

[Thi08] R.J. Thibaux. *Nonparametric Bayesian Models for Machine Learning*. PhD thesis, University of California at Berkeley, 2008.

[Tit08] M. Titsias. The Infinite Gamma-Poisson Model. In *NIPS*, 2008.

[TJBB06] Y. W. Teh, M. I. Jordan, M. J. Beal, and D. M. Blei. Hierarchical Dirichlet processes. *Journal of the American Statistical Association*, 101(476):1566–1581, 2006.

[TK00] S. Tong and D. Koller. Restricted Bayes optimal classifiers. In *Proc. 17th AAAI Conference*, 2000.

[TKW07] Y. W. Teh, K. Kurihara, and M. Welling. Collapsed variational inference for HDP. In *NIPS*, 2007.

[Uhl12] C. Uhler. Geometry of maximum likelihood estimation in Gaussian graphical models. *Annals of Statistics*, 40:238–261, 2012.

[WI98] R. Wolpert and K. Ickstadt. Simulation of Lévy Random Fields. In *Practical Nonparametric and Semiparametric Bayesian Statistics*. Springer-Verlag, 1998.

[WPB11] C. Wang, J. Paisley, and D. Blei. Online variational inference for the hierarchical Dirichlet process. In *AISTATS*, 2011.

[WT90] G. C. G. Wei and M. A. Tanner. A Monte Carlo Implementation of the EM Algorithm and the Poor Man's Data Augmentation Algorithms. *Journal of the American Statistical Association*, 85:699–704, 1990.

[WT11] M. Welling and Y. W. Teh. Bayesian Learning via Stochastic Gradient Langevin Dynamics. In *ICML*, 2011.

[YA06] L. Yin and P. Ao. Existence and Construction of Dynamical Potential in Nonequilibrium Processes without Detailed Balance. *Journal of Physics A: Mathematical and General*, 39(27):8593, 2006.

[ZC12] M. Zhou and L. Carin. Augment-and-conquer negative binomial processes. In *NIPS*, 2012.

[ZC15] M. Zhou and L. Carin. Negative Binomial Process Count and Mixture Modeling. *IEEE Trans. Pattern Anal. Mach. Intell.*, 37(2):307–320, 2015.

[ZHD+01] H. Zha, X. He, C. Ding, H. Simon, and M. Gu. Spectral Relaxation for K-means Clustering. In *NIPS*, 2001.

[ZHDC12] M. Zhou, L. Hannah, D. Dunson, and L. Carin. Beta-negative binomial process and Poisson factor analysis. In *AISTATS*, 2012.

[ZRS16] H. Zhang, S. J. Reddi, and S. Sra. Fast stochastic optimization on Riemannian manifolds. In *NIPS*, 2016.

[ZS16] H. Zhang and S. Sra. First-order methods for geodesically convex optimization. In *COLT*, 2016.