UNIVERSITY OF CALIFORNIA,
IRVINE

Scalable Approximate Bayesian Inference

DISSERTATION

submitted in partial satisfaction of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

in Statistics

by

Tian Chen

Dissertation Committee:
Associate Professor Babak Shahbaba, Chair
Professor Michele Guindani
Assistant Professor Weining Shen

2019

DEDICATION

To my husband *Gordon*

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

ACKNOWLEDGMENTS

First of all, I would like to express my sincere gratitude to my Ph.D. advisor Professor Babak Shahbaba, without whom this dissertation would never have been possible. Professor Shahbaba provided me tremendous help at each step of doing research during my Ph.D. years, from exposing me to interesting topics, providing me valuable research ideas, to giving me constructive feedbacks about research and presentations. With his unconditional support and patient guidance, I was pushed beyond my limits and achieved what I never thought I would have achieved.

I also would like to thank my defense committee members Professor Michele Guindani, Professor Weining Shen, and my advancement committee members Professor Jeffrey Streets, Professor Wesley Johnson, Professor Yaming Yu and Professor Hongkai Zhao, who provided insightful feedbacks about my work.

I would like to express my special thanks to Professor Jeffrey Streets, who educated me about differential geometry and developed the geometric framework for Chapter 2 of this dissertation. In addition, I want to thank Professor Shiwei Lan and Dr. Alexander Vandenberg-Rose for spending their precious time sharing with me their knowledge and giving me valuable advice. Dr. Alexander Vandenberg-Rose also developed the data analysis framework for the work in the Appendix. A special thanks to Fortin's lab for providing data for the work in Chapter 4 and Appendix.

This dissertation is supported by NSF grants DMS 1622490 and IIS 1216045.

CURRICULUM VITAE

Tian Chen

EDUCATION

Doctor of Philosophy in Statistics **2019**
University of California, Irvine *Irvine, CA*

Bachelor of Science in Statistics **2011**
East China Normal University *Shanghai, China*

RESEARCH EXPERIENCE

Graduate Research Assistant **2013–2019**
University of California, Irvine *Irvine, CA*

TEACHING EXPERIENCE

Teaching Assistant **2014–2016**
University of California, Irvine *Irvine, CA*

ABSTRACT OF THE DISSERTATION

Scalable Approximate Bayesian Inference

By

Tian Chen

Doctor of Philosophy in Statistics

University of California, Irvine, 2019

Associate Professor Babak Shahbaba, Chair

The availability of massive computational resources has led to a wide-spread application and development of Bayesian methods. However, in recent years, due to the explosive growth of data volume, developing advanced Bayesian methods for large-scale problems is still a very active area of research. This dissertation is an effort to develop more scalable computational tools for Bayesian inference in big data problems.

At its core, Bayesian inference involves evaluating high dimensional integrals with respect to the posterior distribution of model parameters and/or latent variables. However, the integration does not have closed form in general, and approximation methods are usually the only feasible option. Approximation can be divided into two main categories: deterministic approximation based on variational optimization, and stochastic approximation based on sampling methods.

We start with developing a new variational framework — geometric approximation of posterior (GAP) — based on ambient Fisher geometry. As a variational method, GAP has the potential to scale well to large problems compared to computationally expensive sampling methods. It not only has a well-established mathematical basis — information geometry, but also works as a better alternative to other variational methods such as variational free energy and expectation propagation under certain scenarios.

Next, we focus on another class of approximation scheme based on MCMC sampling. Our method combines auto-encoders with Hamiltonian Monte Carlo (HMC). While HMC is efficient in exploring parameter space with high dimension or complicated geometry, it is computationally demanding since it has to evaluate additional geometric information of the parameter space. Our proposed method, Auto-encoding HMC, is designed to simulate Hamiltonian dynamics in a latent space with a much lower dimension, while still maintaining efficient exploration of the original space. Our method achieves a good balance between efficiency and accuracy for high-dimensional problems.

Besides our work on scalable approximation methods for Bayesian inference, we have also developed a variational auto-encoder (VAE) model based on determinantal point process (DPP) for big data classification problems with imbalanced classes. VAE is a generative model based on variational Bayes and is typically applied to high-dimensional data such as images and texts. In the presence of imbalanced data, our method balances the latent space by using a DPP prior to up-weight the minor classes. We successfully applied our method, henceforth called DPP-VAE, to neural data classification and hand-written digits generation, which are both high-dimensional in nature. Our method provides better results compared to standard VAE when datasets have imbalanced classes.

Chapter 1

Introduction

1.1 Background: Approximate Bayesian Inference

In probabilistic models, there are generally two sets of random variables: observable variables and latent variables. The probabilistic distribution associated with the latter one is referred to as posterior distribution, which is conditional on the observed data. Particularly, in Bayesian statistics, parameters are treated as such latent variables, and the central goal of Bayesian inference is to evaluate integrals with respect to the posterior distribution of the parameters.

More formally, we denote Z to be the parameters or latent variables of interest, and X to be the observable variables. Given the prior information of Z, which is represented by a prior distribution $p_Z(z)$, and the underlying data generating scheme, which is represented by a likelihood function $p_{X|Z}(x|z)$, we have the posterior distribution of Z defined as below:

$$p_{Z|X}(z|x) = \frac{p_Z(z)p_{X|Z}(x|z)}{\int p_Z(z)p_{X|Z}(x|z)dz} \tag{1.1}$$

The task is to evaluate the integral:

$$E_{Z|X}(f(z)) = \int p_{Z|X}(z|x)f(z)dz \tag{1.2}$$

where $f(z)$ is a function of interest of the latent variables.

However, in many cases, the integration does not have an analytically solvable form and hence is infeasible to calculate. Therefore, in practice, a realistic goal is to obtain good approximation to the integrals.

There are two major branches of approximation methods: deterministic approximation and stochastic approximation. Theoretically, stochastic approximation based on sampling methods are guaranteed to generate exact samples given unlimited computational resources. Consequently, a good approximation based on sampling is always computationally expensive. Deterministic approximation, on the other hand, is less computationally demanding and thus scales better to large problems, but is at the compromise of less accuracy.

Important deterministic approximation methods include Laplace's approximation and variational Bayes. As for sampling methods, the most popular algorithms include rejection sampling, importance sampling, and Markov chain Monte Carlo (MCMC). In the next session, we will mainly review the two most commonly used methods — variational Bayes and Markov chain Monte Carlo, respectively.

1.1.1 Variational Bayes

Variational Bayes originates from the idea of optimizing a functional, which is defined to be a mapping from functions to a quantity. The optimization can be simplified by restricting the form of the candidate functions. For example, we can assume that the functions are factorizable.

Given that we are interested in approximating a posterior distribution, the main idea of variational Bayesian inference is to restrict the approximating distributions $q(z) \in Q$, where Q is a family of candidate densities. The goal is to find an optimal density q^* such that the Kullback–Leibler (KL) divergence between the posterior and the approximating density is minimized. KL-divergence is a measure of the difference between two probability densities, which is defined as:

$$KL(p_1 \| p_2) = -\int p_1(x) \log \frac{p_2(x)}{p_1(x)} dx$$

The optimization can be written as follows:

$$q^*(z) = \arg\min_{q(z) \in Q} KL(q(z) \| p_{Z|X}(z|x)) \tag{1.3}$$

Notice that the KL-divergence can be decomposed as below:

$$\begin{aligned} KL(q(z) \| p_{Z|X}(z|x)) &= E_q(\log(q(z))) - E_q(\log(p_{Z|X}(z|x))) \\ &= E_q(\log(q(z))) - E_q(\log(p_{X,Z}(x,z))) + \log p_X(x) \end{aligned}$$

Here, $p_X(x)$ is referred to as the evidence, and the log of evidence is in fact independent of the selection of $q(z)$. To this end, we will not optimize KL-divergence directly since the problem is rarely tractable. Instead, we will maximize the term $-E_q(\log(q(z))) + E_q(\log(p_{X,Z}(x,z)))$ with respect to $q(z)$, which is equivalent to the original optimization goal. The term is introduced as the evidence lower bound (ELBO), because it can be regarded as the lower bound the log of evidence. Notice we have the following decomposition:

$$\log p_X(x) = ELBO(q(z)) + KL(q(z) \| p_{Z|X}(z|x)) \tag{1.4}$$

Given KL-divergence is non-negative, we always have $\log p_X(x) \geq ELBO(q(z))$.

The intuition behind the evidence lower bound is straightforward. We observe that:

$$ELBO(q(z)) = E_q(\log(p_{X|Z}(x|z))) - KL(q(z)\|p_Z(z)) \tag{1.5}$$

The first term indicates maximizing the expected log-likelihood, and the second term encourages the approximating density to match the prior [12].

The most commonly used method to approach this optimization problem is by assuming a mean-field family of distributions and solve it by coordinate ascent algorithm. The mean-field family assumes that $q(z)$ can be factorized to individual $q_j(z_j)$'s with their own variational parameters:

$$q(z) = \prod_{j=1} q_j(z_j|\theta_j)$$

However, the independence assumption will limit the method's ability to capture the covariance structure among latent variables.

With $q(z)$ restricted to belong to the mean-field family, one can show that the optimal density can be found by updating q_j sequentially as follows:

$$q_j(z_j) \propto \exp(E_{-j}(\log p(z_j|z_{-j}, x))) \tag{1.6}$$

which is the exponential of the expected log complete conditional of Z_j with respect to the distribution over other latent variables. The update is terminated when the lower bound converges.

1.1.2 Markov chain Monte Carlo (MCMC)

In MCMC algorithm, we aim at constructing a Markov chain with the posterior $p_{Z|X}(z|x)$ as its stationary distribution. We then collect samples after the chain converges and approximate the integral of interest with the empirical average of the samples:

$$E(f(z)) \simeq \frac{1}{T}\sum_{t=1}^{T} f(z^{(t)})$$

First of all, a Markov chain is a stochastic process which satisfies the Markov property, which states that for a series of random variables $Z^{(1)}, Z^{(2)}, \cdots, Z^{(n)}$, the next state only depends on current state and is independent of all previous states. More formally, for a discrete-time Markov chain, we have:

$$p(z^{(t+1)}|z^{(t)}, \cdots, z^{(1)}) = p(z^{(t+1)}|z^{(t)})$$

To construct a Markov chain such that each state is from the target density $p(z)$, we need to choose a transition probability $T(z'|z)$ to satisfy detailed balance condition:

$$p(z)T(z'|z) = p(z')T(z|z') \tag{1.7}$$

One can show that with this property, the Markov chain will leave the distribution of Z invariant:

$$p(z') = \sum_{z} p(z)T(z'|z) \tag{1.8}$$

Finding such a transition probability is challenging. Metropolis-Hastings algorithm provides a general solution by proposing a state followed by an acceptance step, as shown in Algorithm

1.1. One can easily show that it satisfies detailed balance condition:

Algorithm 1.1 Metropolis-Hastings algorithm

Initialize $z^{(0)}$

for 1 to number of iterations **do**

1. Given current state $z^{(t)}$, generate a proposal $z' \sim q(z'|z^{(t)})$
2. Accept the new proposal z' with a certain probability, which is designed such that the limiting distribution is guaranteed to be $p(z)$. More specifically, take

$$z^{(t+1)} = \begin{cases} z' & \text{with probability } \alpha(z^{(t)}, z') \\ z^{(t)} & \text{with probability } 1 - \alpha(z^{(t)}, z') \end{cases}$$

where $\alpha(x, y) = \min\left\{\frac{p(y)q(x|y)}{p(x)q(y|x)}, 1\right\}$

end for

$$p(z)q(z'|z)\alpha(z, z') = p(z')q(z|z')\alpha(z', z) \tag{1.9}$$

One most popular Metropolis-Hastings algorithm is Gibbs sampling, where the conditional probability $p(z_i|z_{-i})$ can be explicitly obtained for each variable z_i. Then each variable can be sampled sequentially from the conditional probability respectively. Another simple and widely used algorithm is random walk Metropolis, with a Gaussian distribution centered at current state as the transition probability.

In the context of Bayesian statistics, the target density we would like to sample from is the posterior distribution of the parameter $z : p_{Z|X}(z|x)$. The samples can be used to perform Bayesian Inference for z. For example, we can use the mode of the samples as a point estimation for z, which is referred to as maximum a posteriori(MAP) estimation.

1.2 Related Work

Our work mainly focuses on improving advanced methods of Variational Bayes and MCMC algorithm, aiming at developing methods which scales well to large applications. Next, we would like to review some work related to our research.

1.2.1 The Geometry of Statistical Models

Information geometry investigates the differential geometric structure of families of probability distributions [5], which provides a new framework to approach statistical analysis.

Smooth Manifolds

We first discuss the concept of topological manifold [40]. Consider a topological space S. We name it a topological n-manifold if: 1) S is a Hausdorff space: for each pair of points $p, q \in S$, there exist disjoint open subsets $U, V \subset S$ such that $p \in U$ and $q \in V$. 2) S is second countable: there exists a countable basis for the topology of S. 3) S is locally Euclidean of dimension n: every point in the space has a neighborhood which is homeomorphic to an open subset of $\mathbb{R}^n$.

More formally, the third property indicates that for all $p \in S$, we can always find an open subset $U \subseteq S$ which contains p, and an open subset $\tilde{U} \subseteq \mathbb{R}^n$, such that there is a homeomorphism $\varphi : U \to \tilde{U}$. Homeomorphism is a continuous bijective map with continuous inverse. A pair of (U, φ) is called a coordinate chart of Manifold S.

The components we defined so far are not sufficient for calculus on manifolds. We next discuss smooth manifold to make sense of derivatives of functions. Smoothness refers to infinitely differentiable, but it's not a property which is invariant under homeomorphisms.

We therefore need to only focus on "smooth charts". Consider two charts (U, φ) and (V, ψ) of S. If the map $\psi \circ \varphi^{-1} : \varphi(U \cap V) \to \psi(U \cap V)$ is bijective and has a smooth inverse map (diffeomorphism), then these two charts are smoothly compatible.

We could then define a smooth structure on S with a maximal collection (atlas) $\mathcal{A}$ of such charts, i.e., every pair of charts in the atlas are smoothly compatible and the collection is not contained in any larger such collections. We call the pair $(S, \mathcal{A})$ a smooth manifold.

A smooth function is then well-defined on $(S, \mathcal{A})$ —$f : S \to \mathbb{R}$ is smooth if and only if for each coordinate chart (U, φ) in the atlas $\mathcal{A}$, we have $f \circ \varphi^{-1} : \tilde{U} \to \mathbb{R}$ to be smooth.

Statistical Manifolds and Fisher Metric

In the context of statistical distributions, we currently only consider smooth manifold S with a global coordinate system. Suppose we have observations $x = (x_1, \cdots, x_N)$ and an underlying distribution $p(x; \theta)$ which governs the generation of the data. $S = \{p_\theta \equiv p(x; \theta) | \theta = [\theta^1, \cdots, \theta^D] \in \Theta\}$ represents a statistical model consisting of probability distributions $p(x; \theta)$ such that every distribution is uniquely parameterized by a θ. The parameter space Θ is assumed to be an open subset of $\mathcal{R}^n$.

Consider the mapping $\varphi : S \to \Theta$ defined by $\varphi(p_\theta) = \theta$. S then can be regarded as an D-dimensional manifold with a coordinate system θ. We can always conflate the distribution p_θ (a point on manifold) and the parameterization θ (the coordinate) .

Consider the example of Normal distribution:

$$S = \{p(x; \theta) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left\{ - \frac{(x - \mu)^2}{2\sigma^2} \right\} | (\mu, \sigma) \in \Theta\}$$

where $\Theta = \{(\mu, \sigma) | -\infty < \mu < \infty, 0 < \sigma < \infty\}$. Every distribution $p(x; \theta)$ paramerized by a

pair $\theta = (\mu, \sigma)$ is a point on the 2-dimensional manifold S.

Let's denote by $\partial_i = \dfrac{\partial}{\partial \theta^i}$ the tangent vector e_i of the i_{th} coordinate θ^i at point θ. We also denote $l_\theta = \log p(x; \theta)$. Then we can define Fisher information matrix of S at θ as $G(\theta) = [g_{ij}(\theta)]_{D \times D}$ where

$$g_{ij}(\theta) = E_\theta[\partial_i l_\theta \partial_j l_\theta] = \int \partial_i l_\theta \partial_j l_\theta p(x; \theta) dx \tag{1.10}$$

Then by defining the inner product of $[\theta^i]$ as $\langle \partial_i, \partial_j \rangle = g_{ij}(\theta)$, we specify a Riemannian metric in the tangent space T_θ, which is invariant over the choice of coordinate system. We call it the Fisher metric.

1.2.2 Hamiltonian Monte Carlo

In this section, we will use q to represent parameters of interest, to be consistent with conventions in literature.

Compared to random walk Metropolis, Hamiltonian Monte Carlo (HMC) enables more efficient exploration of the parameter space by using geometric information contained in the gradient of the target distribution to inform the proposals.

Hamilton's equations

From a physical perspective, Hamiltonian mechanics is a reformulation of classical mechanics and Lagrangian mechanics, which predicts the same outcomes as these non-Hamiltonian mechanics. Consider a physical system with d degrees of freedom. We denote the position $(q_1, \cdots, q_d) \equiv q$, potential energy $U(q) \equiv U$, momentum $(p_1, \cdots, p_d) \equiv p$, and kinetic energy $K(p) \equiv K$, then we have equivalent equations of motion (i.e. the time evolution of the system

is determined by the equations):

$$\text{Classical:} \quad m\frac{d^2q}{dt^2} = -\frac{\partial}{\partial q}U$$

$$\text{Lagrange:} \quad \frac{d}{dt}\frac{\partial L}{\partial \dot{q}} - \frac{\partial L}{\partial q} = 0$$

$$\text{Hamiltonian:} \quad \frac{dq_i}{dt} = \frac{\partial H}{\partial p_i}, \quad \frac{dp_i}{dt} = -\frac{\partial H}{\partial q_i}$$

where Langrange $L = L(q, \dot{q}, t) = K - U$ and Hamiltonian $H = H(q, p) = U + K$.

In Hamiltonian mechanics, a classical physical system is described by a set of canonical coordinates $z = (q, p)$, which is used to describe a physical system at any given point in time (locating the system within phase space R^{2d}). In the phase space, we can write Hamilton's equations as:

$$\frac{dz}{dt} = J\frac{\partial H}{\partial z}, \text{ where } J = \begin{pmatrix} 0_{d\times d} & I_{d\times d} \\ -I_{d\times d} & 0_{d\times d} \end{pmatrix}$$

We usually use a quadratic kinetic energy: $K(p) = p^T M^{-1} p/2$, which allows the Hamilton's equations to be written as:

$$\frac{dq_i}{dt} = [M^{-1}p]_i \tag{1.11}$$

$$\frac{dp_i}{dt} = -\frac{\partial H}{\partial q_i} \tag{1.12}$$

We can further set the mass matrix to be diagonal.

Hamiltonian dynamics has the following important properties:

- **Reversibility**: if we use the Hamiltonian dynamics to propose in a Metropolis algorithm , this property leaves the desired distribution invariant. More specially, for a one-to-one

mapping T_s from the state at time t to the state at time $t+s$, we have:

$$T_s(q(t), p(t)) = (q(t+s), p(t+s))$$

$$T_s(q(t+s), -p(t+s)) = (q(t), -p(t))$$

- **Conservation of the Hamilton**: we have $\frac{dH}{dt} = 0$, which indicates that theoretically the acceptance rate is one, though this is not the case in practice.

- **Volume preservation**: The phase space distribution $\rho(p, q)$ determines the probability $\rho(p, q)d^n q d^n p$ that the system will be found in the infinitesimal phase space volume $d^n q d^n p$. Liouville's theorem states that the distribution function is constant along any trajectory in phase space. That's saying: $\frac{d\rho}{dt} = 0$. This property holds even when the dynamic is approximated by discretizing time.

- **Symplecticness**: Hamiltonian systems are equipped with a symplectic structure which is preserved by the dynamics. The conservation of the Hamiltonian is only a consequence of this symplectic structure. (This property is stronger than volume preservation when $d > 1$.)

In most cases, we have to discretize time to approximate the Hamilton's equation. It will introduce errors at the same time. A commonly used discretization scheme is leapfrog. One leapfrog iteration is displayed as follows:

$$p_i(t + \frac{\epsilon}{2}) = p_i(t) - \frac{\epsilon}{2}\frac{\partial U}{\partial q_i}(q(t)) \tag{1.13}$$

$$q_i(t + \epsilon) = q_i(t) + \epsilon\frac{p_i(t + \frac{\epsilon}{2})}{m_i} \tag{1.14}$$

$$p_i(t + \epsilon) = p_i(t + \frac{\epsilon}{2}) - \frac{\epsilon}{2}\frac{\partial U}{\partial q_i}(q(t + \epsilon)) \tag{1.15}$$

which provides a proposal from the state at time t for position and momentum variable to

the state at time $t + \epsilon$.

One trick to implement the leapfrog method is to combine the last half step of one iteration with the first half step of the next iteration to be a full step, except for the very first update and the very last update, where we still use half steps for momentum:

$$\begin{cases} p_i(t+\epsilon) = p_i(t+\frac{\epsilon}{2}) - \frac{\epsilon}{2}\frac{\partial U}{\partial q_i}(q(t+\epsilon)) \\ p_i(t+\epsilon+\frac{\epsilon}{2}) = p_i(t+\epsilon) - \frac{\epsilon}{2}\frac{\partial U}{\partial q_i}(q(t+\epsilon)) \end{cases}$$
$$\iff \quad p_i(t+\epsilon+\frac{\epsilon}{2}) = p_i(t+\frac{\epsilon}{2}) - \epsilon\frac{\partial U}{\partial q_i}(q(t+\epsilon))$$

Notice that Leapfrog method preserves phase space volume, because the transformation from $(q(t), p(t))$ to $(q(t), p(t+\epsilon/2))$ is a shear transformation. This only applies to Hamiltonians when there's no interaction between position and momentum, i.e. $H(q,p) = U(q) + K(p)$. Also, leapfrog method is time reversible, simply by applying it with a negative p and then negating p again. Lastly, the leapfrog integrator also preserves the symplectic structure of the Hamiltonian system.

HMC algorithm

Suppose we want to use Metropolis algorithm to sample from a density function $f(q)$ (the space is R^d). In HMC, we use Hamiltonian dynamics to propose a new state for q.

More specifically, we regard the variable of interest q as the position in a Hamiltonian system, and introduce an auxiliary variable p as the momentum variable in this system. We define a joint distribution of q and p which takes a canonical distribution (this concept is adopted

from statistical mechanics):

$$P(q,p) = \frac{1}{Z}\exp(-H(q,p)/T)$$

where T is the temperature of the system, and Z is the normalizing constant. The distribution $f(q)$ and the distribution of momentum p are specified by $H(q,p)$.

If we consider $H(q,p) = U(q)+K(p)$, where q and p are independent, the canonical distribution becomes:

$$P(q,p) = \frac{1}{Z}\exp(-U(q)/T)\exp(-K(p)/T) \propto f(q)g(p)$$

where $f(q)$ is the target density and $g(p)$ is some distribution we define for momentum p.

We set the potential energy $U(q) = -\log(f(q))$. We usually define a zero-mean multivariate Gaussian distribution for p, thus $K(p) = p^T M^{-1} p/2$ with some mass matrix M.

The steps of HMC can be summarized as below:

1. Suppose the current state is $q^{(t)} \equiv q(\tau)$.
2. Resample momentum p from some distribution to be the current $p^{(t)} \equiv p(\tau)$. (We usually use zero-mean Gaussian distribution.)
3. Use leapfrog method to propose a state $(q(\tau+\epsilon), p(\tau+\epsilon))$ from $(q(\tau), p(\tau))$, where ϵ is the step size. (ϵ is a parameter we should tune for good performance of the algorithm.)
4. Repeat 3 for L times. L is also a parameter of the algorithm. So eventually we get the state $(q(\tau+\epsilon L), p(\tau+\epsilon L))$.
5. Set $(q^*, p^*) = (q(\tau+\epsilon L), -p(\tau+\epsilon L))$. (We negate p to guarantee this proposal to be symmetric.)
6. Compute the acceptance rate: $a = min[1, \frac{P(q^*,p^*)}{P(q^{(t)},p^{(t)})}]$. If we set $T=1$ and $H(q,p) =$

$U(q) + K(p)$, we have:

$$\begin{aligned} a &= min[1, \exp(-H(q^*, p^*) + H(q, p))] \\ &= min[1, \exp(-U(q^*) + U(q) - K(p^*) + K(p))] \end{aligned}$$

7. Generate $u \sim U(0, 1)$. We set $q^{(t+1)} = \begin{cases} q^* & u < a \\ q^{(t)} & \text{otherwise} \end{cases}$

We next show a informal proof of the validity of the algorithm. We hope that the chain of q we obtained could finally converges to $f(q)$. Since q and p are independent, and we resample p from the same distribution at each iteration, we just need to show that $P(q, p)$ is the stationary distribution of the Markov chain we obtained by Hamiltonian dynamic proposal. That being said, if the current distribution is $P(\cdot)$, the distribution after one HMC update will remain the same.

We first partition the R^{2d} phase space (q, p) into small regions A_k with the same volume V. $P(A_k)$ is the probability under the canonical distribution. As the volume V goes to zero, we get the corresponding density $P(q, p)$. We use $T(B_k|A_k)$ to represent the mapping from current state A_k to a new state B_k after one HMC iteration, which involves L steps of leapfrog with stepsize ϵ and negation of the momentum variable. Then for all i, j:

$$P(A_i)T(B_j|A_i) = P(B_j)T(A_i|B_j)$$

This is because if $i \neq j$, this equation holds since $T(A_i|B_j) = T(B_j|A_i) = 0$. If $i = j \equiv k$, it's obvious that an equivalent equation holds:

$$V/Z \exp(-H_{A_k}) min[1, \exp(-H_{B_k} + H_{A_k})] = V/Z \exp(-H_{B_k}) min[1, \exp(-H_{A_k} + H_{B_k})]$$

This implies that detailed balance is satisfied and we have:

$$P(B_k)R(B_k) + \sum_i P(A_i)T(B_k|A_i) = P(B_k)$$

where R(X) is the probability that the update for a state in region X leads to rejection of the proposed state. Thus the canonical distribution $P(q, p)$ is the stationary distribution. (Notice $P(B_k) \to P(q, p)$ as $V \to 0$.)

Riemannian HMC

In Riemannian HMC, Fisher Information $G(q)$ is used to adapt the mass Matrix M, which enables more efficient exploration of the parameter space since curvature information is used. An analogy is that for the problem of minimizing a function, Newton's method (second-order, Hessian) is more efficient than gradient descent (first-order). In our case, instead of finding a "optimal" point, we would like to find a "optimal" distribution based on the data, and this is realized by the random kick of momentum p in each iteration. Notice that in Euclidean space, the norm of $\dot{q}$ is $\|\dot{q}\|_M^2 = \dot{q}^T M \dot{q} = p^T M^{-1} p$ (since $dq/dt = M^{-1}p \Rightarrow p = M\dot{q}$) with metric $M = I$, while for a statistical model defined on a Riemannian manifold, $\|\dot{q}\|_{G(q)}^2 = \dot{q}^T G(q) \dot{q} = p^T G^{-1}(q) p$.

1.2.3 Variational Auto-encoder

Variational Auto-encoder (VAE) is a generative model based on variational Bayes and neural networks approximation. It's particularly applicable to high-dimensional data X, such as images and texts. Instead of learning posterior distribution of the latent variables Z, we are more interested in learning the data distribution $p_X(x)$ and generating data from the learned

distribution [20]. Our goal is to maximize the likelihood:

$$p_X(x) = \int p_{X|Z}(x|z)p_Z(z)dz$$

First of all, VAE assumes that $p_{X|Z}(x|z)$ is a distribution which is easy to sample from. If X is continuous, $p_{X|Z}(x|z)$ can be a Gaussian distribution with mean $f(z)$, where $f(z)$ is modeled by a neural network. If X is binary, we can model $p_{X|Z}(x|z)$ by Bernoulli distribution with mean $f(z)$. The prior of Z could simply be a standard normal or other densities which are easy to compute from. In this way, VAE avoids explicitly specifying the latent variables and their correlations, but still capture the latent structure via $f(z)$, in the hope that neural networks can automatically learn the structure given that the model can reproduce X well.

Notice that the induced posterior of Z is also intractable. Similar to variational Bayesian method, VAE uses a simpler distribution $q_Z(z)$ with closed form to approximate the true posterior of Z. Further, it assumes that this simple distribution is conditional on X: $q_Z(z) = q_{Z|X}(z|x)$. In practice, $q_{Z|X}(z|x)$ is usually selected to be $N(z|\mu(x), \Sigma(x))$. $\mu(x)$ and $\Sigma(x)$ are also approximated by neural networks. Then $q_{Z|X}(z|x)$ can be regarded as a 'encoder' which encodes X into Z. Together with $p_{X|Z}(x|z)$ which decodes Z to 'reconstruct' X, the structure resembles a conventional auto-encoder.

The optimization objective of VAE is closely related to variational Bayesian inference. Combining equation (1.4) and (1.5), we have:

$$\log p(x) - KL(q(z|x)\|p(z|x)) = E_q(\log(p(x|z))) - KL(q(z|x)\|p(z)) \tag{1.16}$$

We observe that we can optimize right hand side with respect to neural network parameters of $f(z), \mu(x)$ and $\Sigma(x)$ via back propagation. It's equivalent to maximizing $p_X(x)$ — our ultimate goal, with the bonus of minimizing the KL-divergence between $q_{Z|X}(z|x)$ and the

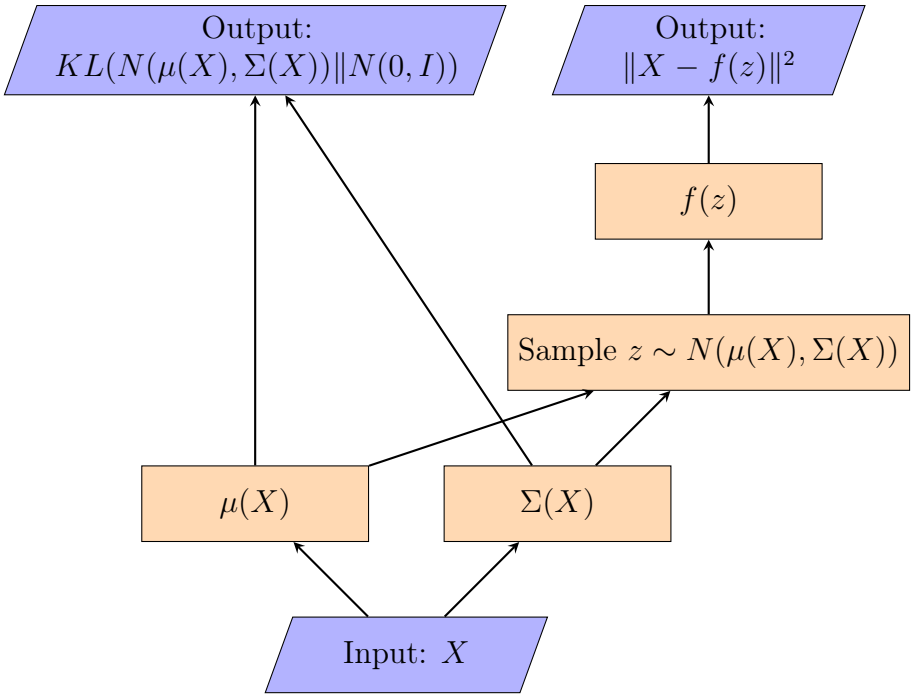


Figure 1.1: Variational Auto-encoder Architecture (X is continuous)

true posterior.

The first term $E_q(\log(p(x|z)))$ can be approximated by Monte Carlo estimate with samples from q. In practice, we implement it stochastically by taking only one sample. When X is continuous and $p_{X|Z}(x|z)$ is Gaussian with mean $f(z)$, we are actually minimizing a "reconstruction loss" $-E_q(\log(p(x|z)))$ which is proportional to $\|X - f(z)\|^2$. The second term has a closed form:

$$KL(N(\mu, \Sigma))\|N(0, I)) = \frac{1}{2}(-\log(\det(\Sigma)) - D + tr(\Sigma) + \mu^T\mu) \tag{1.17}$$

where D is the latent dimension. The network structure and optimization goal is displayed in Figure 1.1.

However, in practice, we need to implement a "reparameterization trick" (Figure 1.2) to make back propagation work. Instead of sampling z from $N(\mu(x), \Sigma(x))$, we first sample ϵ from $N(0, I)$ and construct $z = \mu(x) + \epsilon \cdot \Sigma(x)^{1/2}$. In this way, z becomes a deterministic

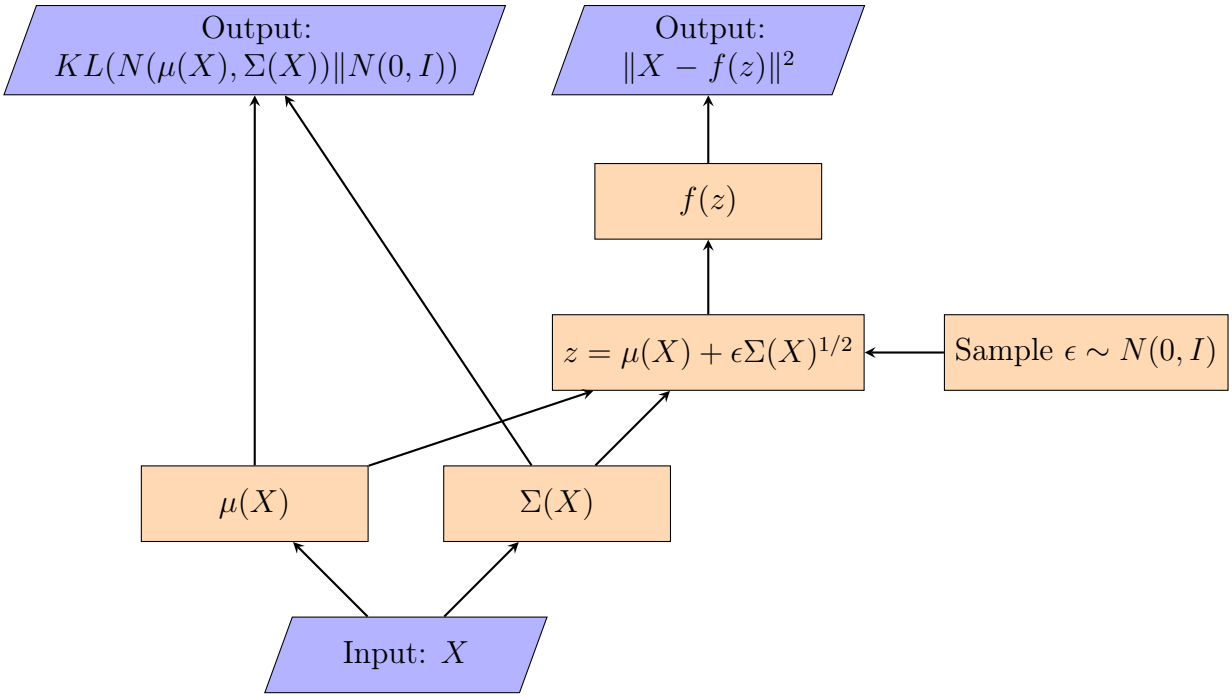


Figure 1.2: Reparameterization Trick (X is continuous)

and continuous function with respect to the neural network parameters, and we could then calculate the gradient.

1.3 Contributions

We now briefly summarize the contributions of this dissertation:

- **A Geometric View of Posterior Approximation** (Chapter 2) provides a novel framework for posterior approximation based on ambient Fisher geometry. As opposed to computationally expensive sampling methods, the method is variational and thus has the potential to scale well to large problems. Instead of optimizing asymmetrical KL-divergence, the objective is a true distance measure derived from a newly constructed geometric structure for the manifold of all probability distributions. The manifold can be regarded as a "unit ball" with infinite dimensions. We name the distance measure

spherical Fisher distance. An optimization algorithm is developed. We showed in our experiments that our method overcomes the shortcomings of other variational methods. For example, unlike variational free energy, our method does not underestimate posterior variance, and unlike variational method based on reverse KL optimization, it does not account for neglectable modes for multi-modal distributions.

- **Speeding Hamiltonian Monte Carlo with Auto-encoders** (Chapter 3) focus on another branch of approximation scheme: MCMC sampling. As discussed in section 1.2.2, HMC is capable of exploring the parameter space more efficiently with distant proposals compared to other MCMC methods, but it is at the cost of spending more computational resource on evaluating geometric information of the space. In the effort of making HMC more scalable to large applications, we developed auto-encoding HMC, which performs dimensionality reduction for the parameters via auto-encoder, and simulates Hamiltonian dynamics in the latent space with a much lower dimension. The proposed latent state is then projected back to the original space as a new proposal. The method is less computationally demanding, while still making efficient proposals in the original space. Our method achieves a good balance between efficiency and accuracy, which is supported by our empirical experiments of high-dimensional simulated data classification and text classification.

- **Determinantal Point Processes as Balancing Priors for Variational Auto-encoder** (Chapter 4) is a work which contributes to improving variational auto-encoder (VAE), a generative model based on variational Bayes, for class imbalance problem. Variational auto-encoder is usually applied to high-dimensional data such as images and texts. In the presence of imbalanced data, the latent space is dominated by the major classes. We correct this by applying a non-parametric prior —- determinantal point process — to up-weight the minor classes. This is due to the property of determinantal point process that it assigns higher probability to dissimilar data points, while we assume

that data from the same class are more similar. The latent representation learned by our method leads to more accurate classification results compared to standard VAE, which uses standard normal as the prior. Our method also is capable of generating more balanced synthetic data given that the training data is extremely imbalanced.

Chapter 2

A Geometric View of Posterior Approximation

Although Bayesian methods are robust and principled, their application in practice could be limited since they typically rely on computationally intensive Markov Chain Monte Carlo algorithms for their implementation. One possible solution is to find a fast approximation of posterior distribution and use it for statistical inference. For commonly used approximation methods, such as Laplace and variational free energy, the objective is mainly defined in terms of computational convenience as opposed to a true distance measure between the target and approximating distributions. In this paper, we provide a geometric view of posterior approximation based on a valid distance measure derived from ambient Fisher geometry. Our proposed framework is easily generalizable and can inspire a new class of methods for approximate Bayesian inference.

2.1 Introduction

In this paper, we are interested in approximating $p_{Z|X}(z|x)$, where Z denotes model parameters or latent variables with prior distribution $p_Z(z)$ and X denotes the observed data [11]. Inference regarding Z typically involves integrating functions over the posterior,

$$E_{Z|X}(g(z)) = \int g(z)p_{Z|X}(z|x)dz \tag{2.1}$$

For instance, $g(z) = z$ if we are interested in estimating the posterior mean. Unfortunately, the integration problem in Bayesian inference is not analytically tractable in most cases. To address this issue, we could use Markov Chain Monte Carlo (MCMC) algorithms by simulating large samples from intractable posterior distributions and using these samples to approximate the above integral. However, MCMC algorithms tend to be computationally intensive, especially for large scale problems. Although many methods have been proposed in recent years to improve computational efficiency of MCMC algorithms (see for example, [49, 48, 28, 47, 55, 58, 29, 61, 19, 13, 52, 51, 50, 7, 46, 6, 38, 15, 16, 62, 27, 57, 56, 63, 66, 2, 30, 32, 59, 9, 14, 39]), extending these methods to high dimensional and complex distributions remains a challenge.

Here, we focus on an alternative family of methods based on deterministic approximation of posterior distribution to cope with intractable problems in Bayesian inference. These methods aim at finding an approximate, but tractable, distribution to replace the exact posterior distribution in order to make statistical inference easier. For example, Laplace's method approximates posterior distribution by a Gaussian distribution with the mean set at the mode of the posterior distribution and the covariance set to the second derivative of the log posterior density evaluated at the mode. While this approach is quite easy to implement, in most cases it could only provide good local approximation around the mode; that is, it

could fail to capture global features of the posterior distribution [11]

An alternative approach is to use variational Bayes methods that approximate the posterior distribution by a much simpler distribution, $p'_Z(z)$, which is assumed to belong to a specific family of models. A divergence is then specified to quantify the dissimilarity between $p_{Z|X}(z|x)$ and $p'_Z(z)$. An optimal $p'_Z(z)$ is chosen from the family of valid distributions by minimizing the divergence.

The variational free energy method (VFE) developed by Feynman and Bogoliubov [42] uses the relative entropy, usually referred to as Kullback-Leibler divergence (KL-divergence), as the measure of dissimilarity between the target and approximating distribution. Consider the following decomposition of the log marginal likelihood:

$$\begin{aligned} \log p_X(x) &= \int p'_Z(z) \log \frac{p_{X,Z}(x,z)}{p'_Z(z)} dz - \int p'_Z(z) \log \frac{p_{Z|X}(z|x)}{p'_Z(z)} dz \qquad (2.2) \\ &= \mathcal{L}(p'_Z) + KL(p'_Z \| p_{Z|X}) \qquad (2.3) \end{aligned}$$

Because KL divergence is non-negative, $\mathcal{L}(p'_Z)$ serves as a lower bound for $\log p_X$. $\mathcal{L}(p'_Z)$ is often referred to as the (negative) variational free energy. Because $\log p_X$ is fixed with respect to p'_Z, minimizing KL-divergence is equivalent to maximizing the lower bound $\mathcal{L}(p'_Z)$; that is, the optimal approximation distribution can be obtained by

$$p'^*_Z = \arg \min_{p'_Z \in P'} KL(p'_Z \| p_{Z|X}) = \arg \max_{p'_Z \in P'} \mathcal{L}(p'_Z) \qquad (2.4)$$

where P' is the set of all valid densities. The purpose of restricting p'_Z to P' is to make the integration over p'_Z tractable and to simplify the optimization problem. In practice, it is common to assume that $p'_Z(z)$ is factorizable.

Note that KL-divergence is not symmetric in general: $KL(p'\|p)$ is not the same as its reverse $KL(p\|p')$. While methods based on variational free energy typically use $KL(p'\|p)$, an alternative method, known as "expectation propagation" [45], uses the reverse KL: $KL(p\|p')$. In this case, by restricting $p'_Z(z)$ to the exponential family, minimization of the reverse KL is simply a moment matching algorithm; that is, by setting the expectation of sufficient statistics of p'_Z equal to that of $p_{Z|X}$, we minimize the reverse KL-divergence. However, the results from direct optimization can be highly inaccurate. Expectation propagation views the joint distribution as a product of factors: $p_{X,Z}(x,z) = \prod_{i=0}^{n} t_i(z)$ where $t_0(z)$ represents the prior and $t_i(z)$ corresponds to the likelihood of data point x_i. The approximating distribution $p'_Z(z)$ is then also assumed to be factorizable: $p'_Z(z)) \propto \prod_{i=0}^{n} \tilde{t}_i(z)$. Each $\tilde{t}_i(z)$ represents an approximating function of $t_i(z)$. The algorithm starts by initializing $\tilde{t}_i(z)$ for $i = 1, \ldots, n$. Then, given $\tilde{t}_{i\neq j}(z)$, each $\tilde{t}_j(z)$ is updated iteratively by moment matching between $p'_Z(z)$ and $t_j(z)\prod_{i\neq j}\tilde{t}_i(z)$.

It is worth noting that above divergence measures are special cases of a family of divergence measure known as α-divergence ([67], [44]),

$$D_\alpha(p\|p') = \frac{\int \alpha p(x) + (1-\alpha)p'(x) - p(x)^\alpha p'(x)^{1-\alpha} dx}{\alpha(1-\alpha)}, \quad \alpha \in (-\infty, \infty) \tag{2.5}$$

The variational free energy method is a special case when $\alpha \to 0$ so we have $\lim_{\alpha\to 0} D_\alpha(p\|p') = KL(p'\|p)$, while the reverse KL divergence corresponds to $\lim_{\alpha\to 1} D_\alpha(p\|p') = KL(p\|p')$. Setting $\alpha = 0.5$, we obtain a symmetric measure $D_{0.5}(p\|p') = 2\int(\sqrt{p(x)} - \sqrt{p'(x)})^2 dx = 4H^2(p,p')$ where $H(p,p')$ represents the Hellinger distance defined as

$$H(p,p') = \left(\frac{1}{2}\int(\sqrt{p(x)} - \sqrt{p'(x)})^2 dx\right)^{\frac{1}{2}} = \frac{1}{\sqrt{2}}\|\sqrt{p(x)} - \sqrt{p'(x)}\|_2 \tag{2.6}$$

Note that α-divergence is not symmetric except for the case of the Hellinger distance ($\alpha = 0.5$), which is rarely used in variational Bayes methods. Most existing variational methods, such as variational free energy and expectation propagation, do not use a real metric for quantifying the approximation error. In contrast to these existing methods, our proposed method, called Geometric Approximation of Posterior (GAP), is based on the *ambient Fisher information metric* that uses a true distance measure, which we call *spherical Fisher distance*. Theoretically, this method provides a novel view of approximate Bayesian inference from the perspective of statistical geometry. Practically, it is a promising method that has the potential to overcome the shortcomings of existing methods. More specifically, unlike MCMC methods, our method does not require computationally intensive simulations. Compared to existing approximation methods, it relies on a true metric and is more flexible in terms of defining the approximating family of distributions.

This paper is organized as follows. In the following section, we present our method based on the spherical Fisher distance. In Section 2.3, we illustrate this approach using simple examples. Finally, we discuss several future directions in Section 2.4.

2.2 Methods

In this section, we present our geometry-based method for approximating posterior distributions. First, we provide a brief overview on geometry of statistical models in general. Next, we discuss "Ambient" Fisher geometry (AFG), which is a particular view of statistical models first observed by [18] (cf. [4**?**]), but has remained relatively unknown in the statistics community. Finally, we show how this geometric view of statistical models can be used to approximate posterior distributions.

2.2.1 The Fisher Metric

Let (M, g) be a smooth Riemannian manifold and let $\mathcal{P}$ denote the space of probability distributions on M. We will use the volume form dV_g to identify distributions with smooth functions which integrate to 1 against this volume form. We can interpret a model Θ as a map from an open set in some parameter space $U \subset \mathbb{R}^D$to $\mathcal{P}$, i.e.

$$\Theta : U \to \mathcal{P}, \qquad (\theta_1, \ldots, \theta_D) \to p_\theta.$$

Denote the associated set of distributions by $S = \{p_\theta | \theta = [\theta_1, \cdots, \theta_D]\}$, which lies in L^1 space and is a subset of $\mathcal{P}$. S is often regarded as an D-dimensional manifold endowed with a Riemannian metric using Fisher information matrix. By introducing a Riemannian metric (i.e. a local inner product on the tangent space at each point) on the manifold, we can derive many geometric notions such as length of curves, geodesic and distance. The Fisher metric is a Riemannian metric defined by Fisher information matrix:

$$g_F^\Theta \left(\frac{\partial}{\partial \theta_i}, \frac{\partial}{\partial \theta_j} \right)_p = \int_M \left(\partial_{\theta_i} \log p_\theta \right) \left(\partial_{\theta_j} \log p_\theta \right) p_\theta dV_g.$$

where $\frac{\partial}{\partial \theta_i}, \frac{\partial}{\partial \theta_j}$ are the i^{th} and j^{th} basis vectors of the tangent space at point p_θ.

2.2.2 Ambient Fisher geometry

In what follows we give a brief summary of "Ambient" Fisher geometry (AFG). This point of view has appeared in the literature, although is not very well-known. Our particular viewpoint was first observed in [18] (cf. [4**?**]).

The Fisher information metric can be interpreted as the Riemannian metric induced by an ambient metric on the infinite dimensional manifold $\mathcal{P}$. To do this we observe that for a

given $p \in \mathcal{P}$, the tangent space can be identified with

$$T_p\mathcal{P} := \left\{ \phi \in C^\infty(M) \mid \int_M \phi dV_g = 0 \right\},$$

which arises by differentiating the unit mass condition on probability distributions. We can then define the ambient Fisher metric on $\mathcal{P}(X)$ by

$$g_F^{\mathcal{P}}(\phi, \psi)_p := \int_M \frac{\phi\psi}{p} dV_g. \tag{2.7}$$

A direct calculation shows that for a model $\Theta : U \to \mathcal{P}$, we have $\Theta^* g_F^{\mathcal{P}} = g_F^{\Theta}$. In other words, the Riemannian geometry induced by the ambient metric on the image of the embedding of the model into $\mathcal{P}$ is the usual Fisher metric.

Our goal is to use the ambient geometric structure of $\mathcal{P}$ to better understand properties of specific models Θ. As it turns out, many geometric properties become clearer when one changes point of view and interprets probability distributions as the unit sphere in the L^2 metric as opposed to the L^1 metric. Specifically, let $\mathcal{Q} = \{q : M \to \mathbb{R} | \int_M q^2 dV_g = 1\}$. We can endow the space of L^2 functions on M with the usual flat inner product, although now interpreted as a Riemannian metric. This induces an inner product on $\mathcal{Q}$, called $g_F^{\mathcal{Q}}$, which is in direct analogy with the geometry inherited by the unit sphere in an ambient Euclidean space. Moreover, direct calculations show that the map $\mathcal{S} : \mathcal{Q} \to \mathcal{P}$ defined by $\mathcal{S}(q) = q^2$ is a Riemannian isometry, i.e. $\mathcal{S}^* g_F^{\mathcal{P}} = g_F^{\mathcal{Q}}$. Thus it is equivalent to work in the space $\mathcal{Q}$ instead of $\mathcal{P}$, which we will now do exclusively.

Using the picture of $\mathcal{Q}$ as the unit sphere of the space of L^2 functions, we can formally derive many basic equations which are fundamental in understanding the ambient Fisher geometry. For instance, we can explicitly solve for geodesics in $\mathcal{Q}$. First, given $q_0 \in \mathcal{Q}$ and $f \in T_q\mathcal{Q}$ a unit tangent vector, the geodesic with initial value q_0 and initial unit norm velocity f exists

on $(-\infty, \infty)$ and takes the form

$$q_t = q_0 \cos t + f \sin t. \tag{2.8}$$

The obvious 2π-periodicity is no surprise, as this curve corresponds to a great circle in the infinite dimensional sphere $\mathcal{Q}$. Also, given $q, q' \in \mathcal{Q}$, the geodesic connecting them takes the form

$$q_t = q \cos t + \frac{q' - q \langle q', q \rangle}{|q' - q \langle q', q \rangle|} \sin t \tag{2.9}$$

Observe that this is well-defined if and only if $q' \neq \pm q$. This makes sense as there is no canonical direction to point in to head from the north pole to the south pole. In this exceptional case one can obtain a geodesic connecting q and q' by choosing an arbitrary initial velocity f and using (2.8). Moreover, a direct integration using (2.9) shows that the distance between two point q, q' is the "arccosine" distance, i.e.

$$d_F^{\mathcal{Q}}(q, q') = \arccos \int_M q q' dV_g, \tag{2.10}$$

which we refer to as spherical Fisher distance in this paper. But since the map $\mathcal{S} : \mathcal{Q} \to \mathcal{P}$ is an isometry, we have the distance between two distributions $p = q^2$, $p' = q'^2 \in \mathcal{P}$ defined as:

$$d_{SF}(p, p') \equiv d_F^{\mathcal{P}}(p, p') = d_F^{\mathcal{Q}}(q, q') = \arccos \int_M \sqrt{p p'} dV_g \tag{2.11}$$

Notice that the distance associated with the usual flat inner product in the ambient "Euclidean

space" (i.e. the space of L^2 functions) is:

$$d_H(p, p') = \left(\int_M (\sqrt{p} - \sqrt{p'})^2 dV_g \right)^{\frac{1}{2}}, \tag{2.12}$$

which is directly related to the Hellinger distance. In contrast, spherical Fisher distance is the distance associated with the inner product on the "unit sphere" manifold $\mathcal{Q}$ (the space of square roots of probability distributions) induced by the usual flat inner product. Although the metric used in our method is different from the Hellinger distance, the two metrics are related in that minimizing spherical Fisher distance is equivalent to minimizing the Hellinger distance between the target and approximating distributions. Geometrically, however, using the spherical Fisher distance is more justifiable and can be optimized more smoothly.

2.2.3 Variational Bayes using AFG

In spite of the difficulty in visualizing a class of distributions (e.g., normal distribution) on the "unit sphere" $\mathcal{Q}$, we could still make use of this idea to approximate complicated distributions through variational methods: after we specify a class of distributions, our task is to find a member of this family with the shortest distance to the target distribution (e.g., posterior distribution). That is, we approximate the target distribution by $p'_Z(z)$, i.e., a member of the assumed family of distributions, by minimizing the spherical Fisher distance to $p_Z(z)$. Notice that unlike KL, the spherical Fisher distance used in our method is based on a true metric. In what follows, we illustrate this idea using a simple problem with analytical solution.

Consider a Gaussian model, $x \sim N(\mu, \tau^{-1})$, with unknown mean, μ, and variance, τ^{-1} (here, τ is the precision parameter). Although the posterior distribution is not tractable in general, it is possible to simplify the problem and find an analytical form for the posterior distribution

by connecting the prior variance of μ to the variance of data as follows:

$$\begin{aligned} \text{Prior:} \quad \mu|\tau &\sim N(\mu_0, (\lambda_0\tau)^{-1}) \\ \tau &\sim Gamma(\alpha_0, \beta_0) \end{aligned}$$

This prior is known as the Normal-Gamma distribution. In this case, given n observed values for x, the posterior distribution has a closed form:

$$(\mu, \tau|x) \sim N(\mu_N^*, (\lambda_N^*\tau)^{-1})Gamma(\alpha_N^*, \beta_N^*),$$

where

$$\begin{aligned} \mu_N^* &= \frac{n\bar{x} + \lambda_0\mu_0}{n + \lambda_0} \\ \lambda_N^* &= \lambda_0 + n \\ \alpha_N^* &= \alpha_0 + \frac{n}{2} \\ \beta_N^* &= \beta_0 + \frac{S}{2} + \frac{n\lambda_0(\bar{x} - \mu_0)^2}{2(n + \lambda_0)} \end{aligned}$$

In Appendix 2.5.1, we show that if we limit our approximating distributions also to the Normal-Gamma family:

$$\begin{aligned} \mu|\tau &\sim N(\mu_N, (\lambda_N\tau)^{-1}) \\ \tau &\sim Gamma(\alpha_N, \beta_N) \end{aligned}$$

then minimizing spherical Fisher distance with respect to $(\mu_N, \lambda_N, \alpha_N, \beta_N)$ leads to the exact

same posterior distribution shown above. That is, by minimizing the spherical Fisher distance between the true posterior p and the approximating distribution p', the optimal p' is exactly p.

2.2.4 Gradient Descent Algorithm

In general, there is no analytical solution for the optimization problem in our method. To address this issue, we develop a gradient-descent optimization algorithm to minimize the distance function. Suppose p_0 is an intractable target distribution (here, posterior distribution) that we want to approximate using a parametric model from Θ. We start from an arbitrary point $\theta_0 \in \Theta$ and improve the approximation via a modified gradient descent in $\mathcal{Q}$. Note that $\sqrt{p_0} \in \mathcal{Q}$ and the model Θ is naturally embedded in $\mathcal{Q}$. Using (2.10), we can calculate the gradient of the distance function. In particular, given a single parameter family $\theta_t \in \Theta$ with derivative $\dot{\theta}$, a direct calculation shows that the directional derivative takes the following form:

$$\nabla_{\dot{\theta}} d(\theta, \sqrt{p_0}) = -\frac{\left\langle \dot{\theta}, \sqrt{p_0} \right\rangle}{\sqrt{1 - \left\langle \theta, \sqrt{p_0} \right\rangle^2}}. \tag{2.13}$$

Because our possible choices of $\dot{\theta}$ are restricted to $T_\theta\Theta$, it is clear that this directional derivative will be minimized by projecting the vector $\sqrt{p_0}$ onto $T_\theta\Theta$,

$$\text{proj}_{T_\theta\Theta}\sqrt{p_0} \;\; = \;\; \sum w_i \langle w_i, \sqrt{p_0} \rangle,$$

where $\{w_i\}$ are orthonormal basis for $T_\theta\Theta$. Ultimately combining this with (2.13) yields the negative gradient vector

$$\overrightarrow{v_0} = \frac{\text{proj}_{T_\theta\Theta}\sqrt{p_0}}{\sqrt{1-\left\langle\theta,\sqrt{p_0}\right\rangle^2}} = \frac{\sum w_i\left\langle w_i,\sqrt{p_0}\right\rangle}{\sqrt{1-\left\langle\theta,\sqrt{p_0}\right\rangle^2}}. \tag{2.14}$$

Therefore, to find an optimal solution for an arbitrary class of models, Θ, we start from an initial point θ_0 on Θ and follow these steps (Figure 2.1):

Step 1 Given θ_0, compute v_0 as in (2.14).

Step 2 Move from θ_0 to θ_1 guided by $\overrightarrow{v_0}$ while confined to Θ. For this, ideally we could follow the geodesic of Θ with θ_0 as the initial position and $\overrightarrow{v_0}$ as the initial velocity to update the parameters. However, because of the difficulty in obtaining such geodesics in general cases, we can instead follow an approximate path. To this end, we set $\overrightarrow{v_0} = \sum_{i=1}^{D} \alpha_i w_i$ and update the parameters separately in each direction: $\theta_1^i = \theta_0^i + \epsilon\alpha_i$ for $i = 1, \ldots, D$, where ϵ is the step size. Iterate the above steps until the updated values of parameters remain close to the current values (i.e., current negative gradient vector $\overrightarrow{v} \approx 0$) or the distance between the target and approximating distributions falls below a predefined threshold.

We iterate through the above steps to obtain the closest point on Θ to $\sqrt{p_0}$.

2.2.5 Gaussian approximation

As discussed above, in practice we usually define a simple class of models to approximate target distributions. Here, we discuss how any arbitrary target distribution can be approximated by

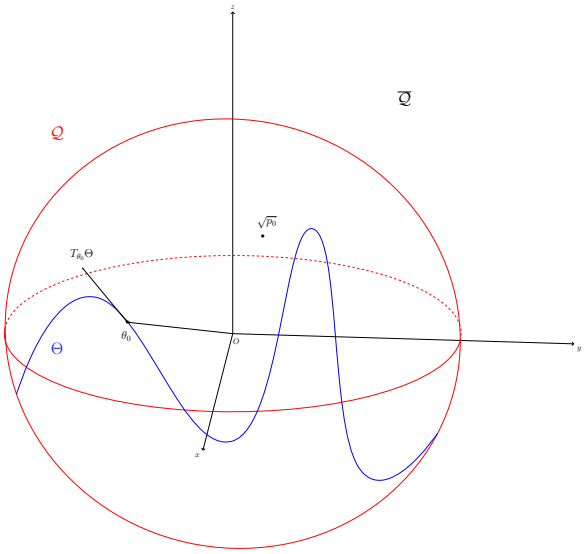


Figure 2.1: A schematic representation of our method. Θ represents the class of approximation distributions. We start from an arbitrary point θ_0 on Θ. Its tangent space with respect to the manifold Θ is denoted as $T_{\theta_0}\Theta$. We move from θ_0 to a new point on Θ directed by the negative gradient vector of the distance function. This is the same direction as the projection of $\sqrt{p_0}$ onto $T_{\theta_0}\Theta$.

a multivariate Gaussian distribution using our method. The resulting algorithm is based on the matrix representation of Gram-Schmidt process. The full details of the procedure can be found in Appendix 2.5.2.

Suppose Θ represents the family of Gaussian models $N(z|\mu, \Sigma)$. Then any point on Θ can be expressed as:

$$p(z|\mu, \Sigma) = (2\pi)^{-\frac{D}{2}} |\Sigma|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(z-\mu)^T \Sigma^{-1}(z-\mu)\right),$$

with the corresponding square root of the density,

$$q(z|\mu, \Sigma) = (2\pi)^{-\frac{D}{4}} |\Sigma|^{-\frac{1}{4}} \exp\left(-\frac{1}{4}(z-\mu)^T \Sigma^{-1}(z-\mu)\right)$$

. Since Σ is constrained to be positive definite, we use its Cholesky decomposition, $\Sigma = LL^T$, and minimize the distance with respect to the lower triangular matrix, L, with unconstrained parameterization. Also, we sometimes express the covariance parameters in a vector form for

simplicity. This is achieved by using the *vech* operator which vectorizes a matrix column-wise, while excluding the upper part of the matrix [22]:$vech(L) = l$.

In order to obtain an orthonormal basis of the tangent space at any point on Θ, consider the push-forwards of basis vectors $\{\frac{\partial}{\partial\theta_i}\}$ with respect to the map from parameter space to root distribution space: $\{\frac{\partial q}{\partial\theta_i}\}$. Thus, starting from an initial point on Θ, $\theta_0 = (\mu_0, L_0) \equiv q(z|\mu_0, \Sigma_0) \equiv q_0$, the orthonormal basis $\{w_i\}$ of the tangent space $T_{\theta_0}\Theta$ can be obtained by orthonormalizing the following basis:

$$
\begin{aligned}
v_\mu &= \frac{\partial q}{\partial \mu}\Big|_{\mu=\mu_0, L=L_0} \\
&= q(z|\mu_0, \Sigma_0)\frac{1}{2}(z-\mu_0)^T\Sigma_0^{-1}, \quad (1 \times D \text{ vector}) \\
v_l &= \frac{\partial q}{\partial vech(L)}\Big|_{\mu=\mu_0, L=L_0} \\
&= q(z|\mu_0, \Sigma_0)\left[-\frac{1}{4}vec(\Sigma_0^{-T})^T + \frac{1}{4}((z-\mu_0)^T \otimes (z-\mu_0)^T)(\Sigma_0^{-T} \otimes \Sigma_0^{-1})\right] \\
&\left[I + T_{D,D}^T - R_D^T\right]\left[(I_D \otimes L_0)T_{D,D} + (L_0 \otimes I_D)\right] S_D^T, \quad (1 \times \frac{D(D+1)}{2} \text{ vector})
\end{aligned}
$$

Given $\{w_i\} \equiv \left(\{w_{\mu_i}\}_{i=1}^{D}, \{w_{li}\}_{i=1}^{\frac{D(D+1)}{2}}\right)$, we have

$$
\overrightarrow{v_0} = \frac{\sum_{i=1}^{D}\langle w_{\mu_i}, \sqrt{p_0}\rangle w_{\mu_i} + \sum_{i=1}^{D(D+1)/2}\langle w_{li}, \sqrt{p_0}\rangle w_{li}}{\sqrt{1 - \langle\theta_0, \sqrt{p_0}\rangle^2}}
$$

Finally, we update the parameters as follows:

$$
\begin{aligned}
\mu_i^{(t+1)} &= \mu_i^{(t)} + \epsilon_{\alpha_i}\langle w_{\mu_i}, \sqrt{p_0}\rangle / \sqrt{1 - \langle\theta_0, \sqrt{p_0}\rangle^2} \quad i = 1, \cdots, D \\
l_i^{(t+1)} &= l_i^{(t)} + \epsilon_{\beta_i}\langle w_{li}, \sqrt{p_0}\rangle / \sqrt{1 - \langle\theta_0, \sqrt{p_0}\rangle^2} \quad i = 1, \cdots, \frac{D(D+1)}{2}
\end{aligned}
$$

Algorithm 2.2 GAP: obtaining $\langle w_{\mu_j}, \sqrt{p_0} \rangle$

Generate iid samples $z^{(t)}, t = 1, \cdots, T$ from $q^2(z|\mu_0, \Sigma_0)$
Calculate a_i: the mean of $\frac{\sqrt{p_0(z^t)}}{q(z^{(t)}|\mu_0,\Sigma_0)} \frac{1}{2} \left[(z^{(t)} - \mu_0)^T \Sigma_0^{-1}\right]_i$, $t = 1, \cdots, T$ for each $i = 1, \cdots, D$
Let $A = \frac{1}{4}\Sigma_0^{-1}$
for $j = 1$ to D **do**
    Obtain A_j as the jth order leading principal submatrix of A
    Calculate D_j as the determinant of A_j
    Calculate $M_{j,i}$ for each $i = 1, \cdots, j$, where $M_{j,i}$ is a minor of A_j
    Calculate $\langle w_{\mu_j}, \sqrt{p_0} \rangle = \frac{1}{\sqrt{D_{j-1} D_j}} \sum_{i=1}^{j} (-1)^{j+i} M_{j,i} a_i$
end for

where $\epsilon_{\alpha_i}, \epsilon_{\beta_i}$ are stepsizes. Algorithm 2.2 and 2.3 show the steps to obtain $\langle w_{\mu_i}, \sqrt{p_0} \rangle$ and $\langle w_{l_i}, \sqrt{p_0} \rangle$ respectively.

2.3 Illustrations

In this section, we evaluate our approximation method using three illustrative examples. We first start with a toy example, where we approximate a t-distribution with a normal distribution. Next, we use our method to find a normal approximation to the posterior distribution of parameters in a Bayesian logistic regression model. Our final example involves approximating a bimodal distribution, which is a mixture of two normals.

2.3.1 A toy example: approximating the $t(1)$ distribution

For our first example, we use our method to find a normal approximation, $N(\mu, \sigma^2)$, to the t-distribution with 1 degree of freedom, $t(1)$. Although this is just a one-dimensional case of the procedure discussed in section 2.2.5, we would like to elaborate it in more details here.

Algorithm 2.3 GAP: obtaining $\langle w_{lj}, \sqrt{p_0} \rangle$

Pre-calculate $T_{D,D}, R_D, S_D$ as defined in Appendix 2.5.2.
Pre-calculate $U_D = I + T_{D,D}^T - R_D^T$.

Calculate $V_D = \left[(I_D \otimes L_0) T_{D,D} + (L_0 \otimes I_D) \right] S_D^T$
Calculate $E(W_D^T W_D)$ (Appendix 2.5.2. First obtain $vec(\Sigma_0)$, Σ_0^{-1}, $vec(\Sigma_0^{-1})$, $\Sigma_0 \otimes \Sigma_0$, $\Sigma_0^{-1} \otimes \Sigma_0^{-1}$. Permute $\Sigma_0 \otimes \Sigma_0$ to obtain $\left[(\Sigma_0 \otimes \Sigma_0)_{ikjl} \right], \left[(\Sigma_0 \otimes \Sigma_0)_{iljk} \right]$)

Calculate $B = V_D^T U_D^T E(W_D^T W_D) U_D V_D$

Generate iid samples $z^{(t)}, t = 1, \cdots, T$ from $q^2(z|\mu_0, \Sigma_0)$
Calculate b_i: the mean of $\frac{\sqrt{p_0(z^t)}}{q(z^{(t)}|\mu_0, \Sigma_0)} [(-\frac{1}{4} vec(\Sigma_0^{-T})^T + \frac{1}{4}((z^{(t)} - \mu_0)^T \otimes (z^{(t)} - \mu_0)^T)(\Sigma_0^{-T} \otimes \Sigma_0^{-1}) U_D V_D]_i$, $t = 1, \cdots, T$ for each $i = 1, \cdots, \frac{D(D+1)}{2}$
for $j = 1$ to $\frac{D(D+1)}{2}$ **do**
  Obtain B_j as the jth order leading principal submatrix of B
  Calculate E_j as the determinant of B_j
  Calculate $N_{j,i}$ for each $i = 1, \cdots, j$, where $N_{j,i}$ is a minor of B_j
  Calculate $\langle w_{l_j}, \sqrt{p_0} \rangle = \frac{1}{\sqrt{E_{j-1} E_j}} \sum_{i=1}^{j} (-1)^{j+i} N_{j,i} b_i$
end for

For this problem, we have

$$\begin{aligned}
&\text{the square root density of } t(1): && \sqrt{p_0(x)} = \pi^{-\frac{1}{2}} (1 + x^2)^{-\frac{1}{2}} \\
&\text{the square root density of } N(\mu, \sigma^2): && \sqrt{p(x|\mu, \sigma^2)} = (2\pi)^{-\frac{1}{4}} (\sigma^2)^{-\frac{1}{4}} \exp(-\frac{1}{4\sigma^2}(x - \mu)^2) \\
& && \equiv q(x|\mu, \sigma^2)
\end{aligned}$$

To obtain unconstrained parameterization, we update σ $(-\infty < \sigma < \infty)$ instead of σ^2. The basis for the tangent space $T_{\theta_0}\Theta$ are as follows:

$$\begin{aligned}
v_\mu = \frac{\partial q}{\partial \mu}\Big|_{\mu=\mu_0, \sigma=\sigma_0} &= q(x|\mu_0, \sigma_0^2) \frac{1}{2\sigma_0^2}(x - \mu_0) \\
v_\sigma = \frac{\partial q}{\partial \sigma}\Big|_{\mu=\mu_0, \sigma=\sigma_0} &= q(x|\mu_0, \sigma_0) \left[-\frac{1}{2}(\sigma_0)^{-1} + \frac{1}{2}\sigma_0^{-3}(x - \mu_0)^2 \right], \text{ where } -\infty < \sigma < \infty
\end{aligned}$$

from which, we obtain an orthonormal basis of $T_{\theta_0}\Theta$,

$$\begin{aligned} w_\mu &= q(x|\mu_0, \sigma_0^2)\frac{1}{\sqrt{\sigma_0^2}}(x-\mu_0) \\ w_\sigma &= q(x|\mu_0, \sigma_0^2)\frac{\sqrt{2}}{2}(\frac{(x-\mu_0)^2}{\sigma_0^2}-1) \end{aligned}$$

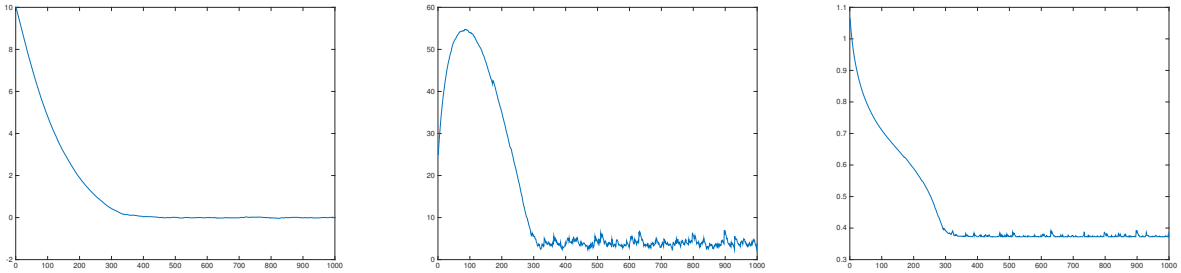
Finally, the negative gradient vector at θ_0 is $\overrightarrow{v_0} = \frac{\langle w_\mu, \sqrt{p_0}\rangle w_\mu + \langle w_\sigma, \sqrt{p_0}\rangle w_\sigma}{\sqrt{1-\langle \theta_0, \sqrt{p_0}\rangle^2}}$, where

$$\begin{aligned} \langle w_\mu, \sqrt{p_0}\rangle &= \left(\frac{\pi}{2\sigma_0^2}\right)^{-\frac{1}{4}}\frac{c_2}{\sqrt{\sigma_0^2}} \\ \langle w_\sigma, \sqrt{p_0}\rangle &= \left(\frac{\pi}{2\sigma_0^2}\right)^{-\frac{1}{4}}(\frac{\sqrt{2}c_1}{2\sigma_0^2}-\frac{\sqrt{2}c_3}{2}) \\ \langle \theta_0, \sqrt{p_0}\rangle &= \left(\frac{\pi}{2\sigma_0^2}\right)^{-\frac{1}{4}}c_3 \end{aligned}$$

Here, c_1, c_2, c_3 are integrals over q_0 and can be expressed as expectations with respect to $q(x|\mu_0, \sigma_0^2)$,

$$\begin{aligned} c_1 &= E_{q_0^2}(\frac{(x-\mu_0)^2}{\sqrt{1+x^2}\exp(-\frac{1}{4\sigma_0^2}(x-\mu_0)^2)}) \\ c_2 &= E_{q_0^2}(\frac{x-\mu_0}{\sqrt{1+x^2}\exp(-\frac{1}{4\sigma_0^2}(x-\mu_0)^2)}) \\ c_3 &= E_{q_0^2}(\frac{1}{\sqrt{1+x^2}\exp(-\frac{1}{4\sigma_0^2}(x-\mu_0)^2)} \end{aligned}$$

We approximate these integrals using the Monte Carlo approximation method.



(a) μ (b) σ^2 (c) Distance

Figure 2.2: Approximating $t(1)$ with $N(\mu, \sigma^2)$. As we can see, the distance reaches its minimum after 400 iterations, where μ and σ^2 converge to 0.0005 and 3.7468 respectively.

Ideally, we can update θ_0 by following the geodesic flow $\gamma(t)$ with $\gamma(0) = \theta_0$ and $\gamma'(0) = \overrightarrow{v_0}$. For simplicity, however, we follow an approximate path and update the parameters as follows:

$$\begin{aligned}
\mu^{(t+1)} &= \mu^{(t)} + \epsilon_\alpha \frac{\langle w_\mu, \sqrt{p_0} \rangle}{\sqrt{1 - \langle \theta_0, \sqrt{p_0} \rangle^2}} \\
\sigma^{(t+1)} &= \sigma^{(t)} + \epsilon_\beta \frac{\langle w_\sigma, \sqrt{p_0} \rangle}{\sqrt{1 - \langle \theta_0, \sqrt{p_0} \rangle^2}} \\
\sigma^{2(t+1)} &= (\sigma^{(t+1)})^2
\end{aligned}$$

See Appendix 2.5.3 for more details.

We initialize $(\mu, \sigma) = (10, 5)$ and set stepsizes $\epsilon_\alpha = 0.1, \epsilon_\beta = 5$. The sequence of parameters and the distance between the target and approximating distributions over 1000 iterations are shown in Figure 2.2. The approximating distribution is converging to $N(0.0005, 3.7468)$ and the distance reaches its minimum after 400 iterations. Note that the stochastic path towards the end is due to the Monte Carlo approximation. The corresponding density functions for the target and approximating distributions are shown in Figure 2.3.

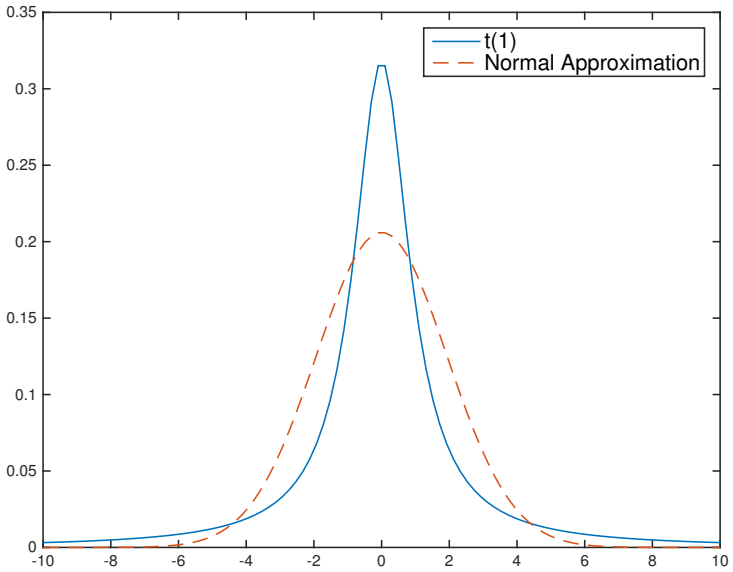


Figure 2.3: Approximating $t(1)$ with $N(0.0005, 3.7468)$ based on our method.

2.3.2 Logistic Regression

For our next example, we consider Bayesian inference based on the following logistic regression model:

$$\begin{aligned} \text{Likelihood:} \quad & y_i|X_i,\beta \sim \text{Bernoulli}(p_i = \frac{e^{X_i^T\beta}}{1+e^{X_i^T\beta}}) \quad i = 1, \cdots, n \\ \text{Prior:} \quad & \beta \sim N_D(\mu_*, \Sigma_*) \end{aligned}$$

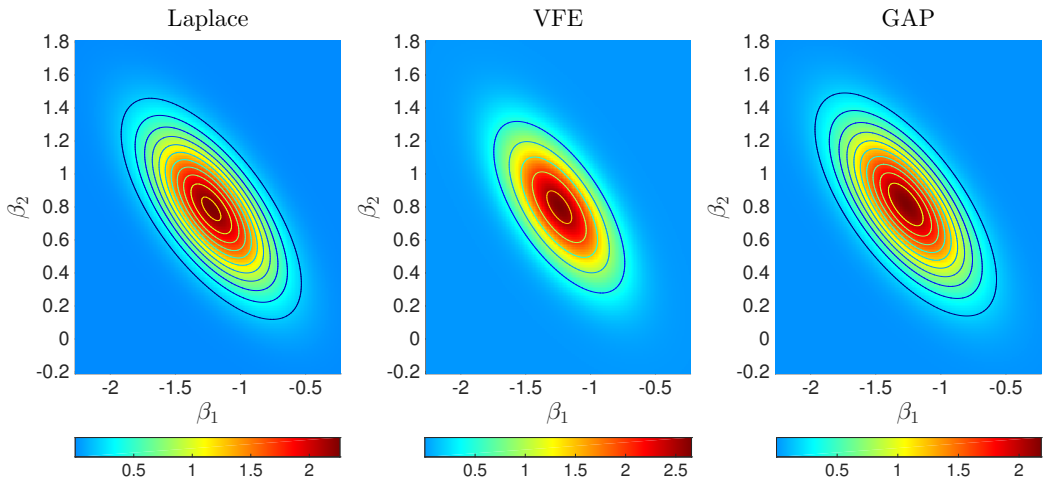


Figure 2.4: Comparing our approximation method (GAP) to Laplace's approximation, and variational free energy (VFE) based on a logistic regression model.

The posterior distribution of model parameters and its square root are

$$
\begin{aligned}
p_0(\beta|X,y) &= p(y|\beta)p(\beta)/p(y) \\
&= \frac{1}{p(y)} \prod_{i=1}^{n} \left(\frac{e^{X_i^T\beta}}{1+e^{X_i^T\beta}} \right)^{y_i} \left(1 - \frac{e^{X_i^T\beta}}{1+e^{X_i^T\beta}} \right)^{1-y_i} \\
&\quad (2\pi)^{-\frac{D}{2}} |\Sigma_*|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\beta-\mu_*)^T {\Sigma_*}^{-1}(\beta-\mu_*) \right) \\
\sqrt{p_0(\beta|X,y)} &= p(y)^{-\frac{1}{2}} \left[\prod_{i=1}^{n} e^{\frac{y_i}{2} X_i^T\beta} (1+e^{X_i^T\beta})^{-\frac{1}{2}} \right] (2\pi)^{-\frac{D}{4}} |\Sigma_*|^{-\frac{1}{4}} \\
&\quad \exp\left(-\frac{1}{4}(\beta-\mu_*)^T {\Sigma_*}^{-1}(\beta-\mu_*) \right)
\end{aligned}
$$

For approximation, we use the family of the D-dimensional Gaussian distributions and implement the algorithm presented in Section 2.2.5. Figure 2.4 shows the normal approximation based on our method along with approximations obtained by Laplace's method and variational free energy (VFE) based on a dataset of size $N = 100$ with $\beta_0 = 0.5, \beta_1 = -1.5$, and $\beta_2 = 1$. For this example, we generated (x_1, x_2) from a bivariate normal distribution with zero means, unit variances and correlation 0.7.

As expected, the approximating distribution based on variational free energy is more compact than the true distribution [42]. Note that here we used a local variational method, where a lower bound is found for a part of the entire probabilistic model to simplify the approximation ([11] [34]). For Bayesian logistic regression, a lower bound $h(\beta, \xi)$ for $p(y|\beta)$ can be derived using the convex duality framework, where ξ are variational parameters. The variational posterior then can be obtained by maximizing $\mathcal{L}(\xi) \equiv \ln \int h(\beta, \xi) p(\beta) d\beta \leqslant \ln \int p(y\beta) p(\beta) d\beta = \ln p(y)$ using the Expectation-Maximization (EM) algorithm.

For this example, our approximating distribution is almost the same as what we obtain from Laplace's method. However, as illustrated by our next example, this is not the case in general.

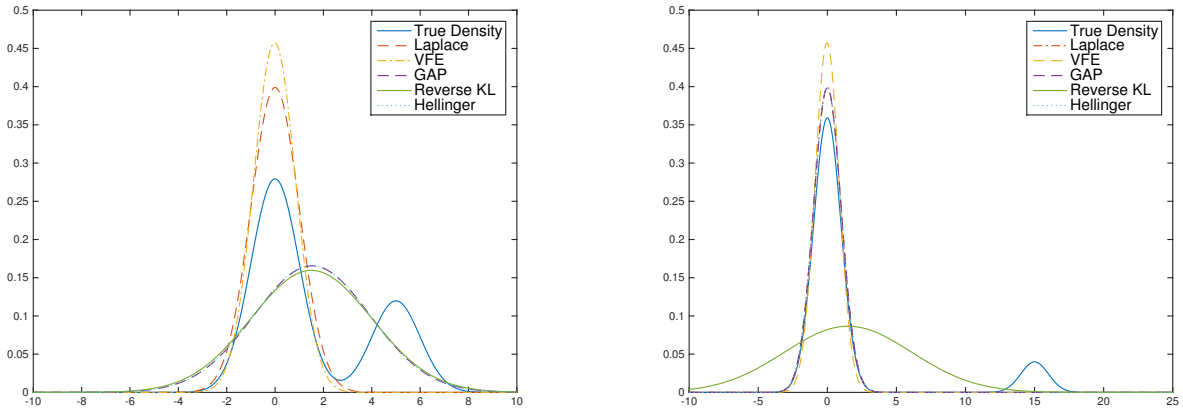
2.3.3 Approximating a bimodal distribution

For our final example, we use our method to find a univariate Gaussian approximation to mixture of normals. First, we use our method to find a normal approximation to the following bimodal distribution:

$$x \sim 0.7N(0, 1) + 0.3N(5, 1).$$

The left panel of Figure 2.5 compares the result of our model to those based on Laplace's approximation and α-divergence, for different values of α (KL-divergence, reverse KL-divergence, and the Hellinger distance). As we can see, while Laplace's approximation and variational free energy (VFE) capture the first mode only (hence, underestimating the variance), our method increases the variance to cover both modes. Similar results are obtained based on reverse KL-divergence and the Hellinger distance. As expected, the results based on GAP and the Hellinger distance are almost indistinguishable.

Recall that $\lim_{\alpha \to 0} D_\alpha(p\|p') = KL(p'\|p)$ and $\lim_{\alpha \to 1} D_\alpha(p\|p') = KL(p\|p')$. When $\alpha < 0$, minimizing $D_\alpha(p\|p')$ tends to give zero-forcing results, because when p is close to zero, p'



(a) $x \sim 0.7N(0,1) + 0.3N(5,1)$ (b) $x \sim 0.9N(0,1) + 0.1N(15,1)$

Figure 2.5: Approximating bimodal distributions using our method (GAP), Laplace's approximation, variational free energy, Reverse KL and the Hellinger distance.

also has to be close to zero to avoid large penalties. Therefore, the VFE method in this case captures a single mode. However, when $\alpha > 1$, the result is zero-avoiding, i.e., p' tends to be greater than zero in regions where p is greater than zero. Thus, results based on reverse KL will average across both modes. When $0 < \alpha < 1$, the results are in between: they are neither zero-forcing nor zero-avoiding, so it tends to cover across modes but will fail to find modes that are far from the main mass [44]. To see this, we consider another mixture distribution which has a mode far from the main mass:

$$x \sim 0.9N(0,1) + 0.1N(15,1).$$

The right panel of Figure 2.5 shows the corresponding results. Here, we observe that the Hellinger distance fails to capture the far mode, as opposed to reverse KL. As discussed before, minimizing spherical Fisher distance is equivalent to minimizing the Hellinger distance, so the approximating distribution based of our method (GAP) is similar to the distribution based on the the Hellinger distance in both examples.

2.4 Discussion

We have proposed a novel framework for approximating posterior distributions and illustrated its performance using several examples. Application of our method, however, can go well beyond what discussed here. As a deterministic approximation approach, our method has the potential to scale better compared to MCMC methods. Compared to other deterministic approaches, our method's flexibility and generalizability could lead to substantially more accurate approximation of posterior distribution, which in turn would lead to more accurate statistical inference.

Although in this paper we limited the class of approximating distributions to normals, our method can be generalized to other approximating distributions as long as we could obtain the orthonormal basis $\{w_i\}$ with respect to each particular point θ on Θ. For example, we can set Θ to be mixture of normals. This would allow for more flexibility in approximating target distributions.

To make our method more practical, we should substantially improve its computational efficiency. Currently, the computational cost of our method is mainly dominated by finding the orthonormal basis of $T_\theta\Theta$. Also, finding alternatives to Monte Carlo method for approximating intractable integrals in our algorithm could help to make our method more efficient.

Finally, we need to further study the properties of our proposed method and its connection to other approximation approaches such as those based on α-divergence. It will also be of great importance to identify classes of approximating distributions that lead to convex optimization problems. For non-convex problems, we need to improve our numerical optimization method to avoid falling into local minima.

2.5 Supplementary

2.5.1 An illustrative example with analytical solution

We now provide the details for the illustrative example with analytical solution discussed in Section 2.2.3. For this problem we have,

Posterior density:

$$\begin{aligned} p &\propto P(x|\mu,\tau)P(\mu|\tau)P(\tau) \\ &= (2\pi)^{-\frac{n}{2}}\tau^{\frac{n}{2}}\exp\left(-\frac{\tau}{2}\sum(x_i-\mu)^2\right)\cdot(2\pi)^{-\frac{1}{2}}(\lambda_0\tau)^{\frac{1}{2}}\exp(-\frac{\lambda_0\tau}{2}(\mu-\mu_0)^2) \\ &\frac{\beta_0^{\alpha_0}}{\Gamma(\alpha_0)}\tau^{\alpha_0-1}\exp(-\beta_0\tau) \end{aligned}$$

Its square root:

$$\begin{aligned} \sqrt{p} \propto (2\pi)^{-\frac{n}{4}}\tau^{\frac{n}{4}}\exp\left(-\frac{\tau}{4}\sum(x_i-\mu)^2\right)\cdot(2\pi)^{-\frac{1}{4}}(\lambda_0\tau)^{\frac{1}{4}}\exp(-\frac{\lambda_0\tau}{4}(\mu-\mu_0)^2) \\ \sqrt{\frac{\beta_0^{\alpha_0}}{\Gamma(\alpha_0)}}\tau^{\frac{\alpha_0-1}{2}}\exp(-\frac{\beta_0\tau}{2}) \end{aligned}$$

Approximating density:

$$\begin{aligned} p' &= P'(\mu|\tau)P'(\tau) \\ &= (2\pi)^{-\frac{1}{2}}(\lambda_N\tau)^{\frac{1}{2}}\exp(-\frac{\lambda_N\tau}{2}(\mu-\mu_N)^2)\frac{\beta_N^{\alpha_N}}{\Gamma(\alpha_N)}\tau^{\alpha_N-1}\exp(-\beta_N\tau) \end{aligned}$$

Its square root:

$$\sqrt{p'} = (2\pi)^{-\frac{1}{4}}(\lambda_N\tau)^{\frac{1}{4}}\exp(-\frac{\lambda_N\tau}{4}(\mu-\mu_N)^2)\sqrt{\frac{\beta_N^{\alpha_N}}{\Gamma(\alpha_N)}}\tau^{\frac{\alpha_N-1}{2}}\exp(-\frac{\beta_N\tau}{2})$$

The spherical Fisher distance between p and p' is

$$d_{SF}(p,p') = \arccos \int\int \sqrt{pp'} d\mu d\tau$$
$$\propto \arccos \int\int f(\mu,\tau) \sqrt{\frac{\beta_N^{\alpha_N}}{\Gamma(\alpha_N)}} \lambda_N^{\frac{1}{4}} \frac{\Gamma(\alpha^*)\sqrt{2\pi}}{\beta^{*\alpha^*} \lambda^{*\frac{1}{2}}} d\mu d\tau$$

where

$$f(\mu,\tau) = \frac{\beta^{*\alpha^*} \lambda^{*\frac{1}{2}}}{\Gamma(\alpha^*)\sqrt{2\pi}} \tau^{\alpha^* - \frac{1}{2}} \exp(-\beta^*\tau) \exp(-\frac{\lambda^*\tau(\mu-\mu^*)^2}{2})$$

is the joint Normal-Gamma density of (μ, τ) parameterized by $(\mu^*, \lambda^*, \alpha^*, \beta^*)$:

$$\mu^* = \frac{n\bar{x} + \lambda_0\mu_0 + \lambda_N\mu_N}{n + \lambda_0 + \lambda_N}$$
$$\lambda^* = \frac{n + \lambda_0 + \lambda_N}{2}$$
$$\alpha^* = \frac{n}{4} + \frac{\alpha_0 + \alpha_N}{2}$$
$$\beta^* = \frac{S}{4} + \frac{\beta_0 + \beta_N}{2} + \frac{n\lambda_0(\bar{x}-\mu_0)^2 + n\lambda_N(\bar{x}-\mu_N)^2 + \lambda_0\lambda_N(\mu_0-\mu_N)^2}{4(n + \lambda_0 + \lambda_N)}$$
$$\text{with } S = \sum (x_i - \bar{x})^2$$

Therefore, we have: $d_{SF}(p,p') \propto \arccos \sqrt{\frac{\beta_N^{\alpha_N}}{\Gamma(\alpha_N)}} \frac{\lambda_N^{\frac{1}{4}} \Gamma(\alpha^*)}{\beta^{*\alpha^*} \lambda^{*\frac{1}{2}}} \equiv \arccos g(\mu_N, \lambda_N, \alpha_N, \beta_N)$.

Since *arccos* function is monotone decreasing and *log* function is monotone increasing, minimizing the spherical Fisher distance between p and p' with respect to $\mu_N, \lambda_N, \alpha_N, \beta_N$ is equivalent to maximizing:

$$\log g(\mu_N, \lambda_N, \alpha_N, \beta_N) = \frac{\alpha_N}{2}\log\beta_N - \frac{1}{2}\log\Gamma(\alpha_N) + \frac{1}{4}\log\lambda_N + \log\Gamma(\alpha^*) - \alpha^*\log\beta^* - \frac{1}{2}\log\lambda^*.$$

To maximize the above function, we need to solve $\frac{\partial \log g}{\partial \mu_N} = 0$, $\frac{\partial \log g}{\partial \lambda_N} = 0$, $\frac{\partial \log g}{\partial \alpha_N} = 0$,

$\frac{\partial \log g}{\partial \beta_N} = 0$. Note that

$$\begin{aligned}
\frac{\partial \alpha *}{\partial \alpha_N} &= \frac{1}{2}, \quad \frac{\partial \beta *}{\partial \beta_N} = \frac{1}{2}, \quad \frac{\partial \lambda^*}{\partial \lambda_N} = \frac{1}{2} \\
\frac{\partial \beta *}{\partial \lambda_N} &= \frac{\left(n(\mu_N - \bar{x}) + \lambda_0(\mu_N - \mu_0)\right)^2}{4(n + \lambda_0 + \lambda_N)^2} \\
\frac{\partial \beta *}{\partial \mu_N} &= \frac{\lambda_N \mu_N (n + \lambda_0) - \lambda_N (n\bar{x} + \lambda_0 \mu_0)}{2(n + \lambda_0 + \lambda_N)}
\end{aligned}$$

To find the optimal μ_N, λ_N, we solve

$$\begin{aligned}
&\frac{\partial \log g}{\partial \mu_N} = 0 \Rightarrow \mu_N = \frac{n\bar{x} + \lambda_0 \mu_0}{n + \lambda_0} \\
&\frac{\partial \log g}{\partial \lambda_N} = 0 \Rightarrow \frac{1}{4\lambda_N} - \frac{1}{4\lambda^*} - \frac{\alpha^*}{\beta^*} \frac{\left(n(\mu_N - \bar{x}) + \lambda_0(\mu_N - \mu_0)\right)^2}{4(n + \lambda_0 + \lambda_N)^2} = 0
\end{aligned}$$

But given $\mu_N = \frac{n\bar{x} + \lambda_0 \mu_0}{n + \lambda_0}$, we have $\lambda_N = \lambda^* \Rightarrow \lambda_N = \lambda_0 + n$

Finally, we find the optimal α_N and β_N as follows:

$$\begin{aligned}
\frac{\partial \log g}{\partial \alpha_N} = 0 \quad &\Rightarrow \quad \log \beta_N - \log \beta^* = \psi(\alpha_N) - \psi(\alpha^*), \text{ where } \psi(x) = \frac{d}{dx} \ln \Gamma(x) = \frac{\Gamma'(x)}{\Gamma(x)} \\
\frac{\partial \log g}{\partial \beta_N} = 0 \quad &\Rightarrow \quad \frac{\beta_N}{\beta^*} - \frac{\alpha_N}{\alpha^*} = 0
\end{aligned}$$

Trivially, we solve $\log \alpha_N - \log \alpha^* = \psi(\alpha_N) - \psi(\alpha^*)$ by setting $\alpha_N = \alpha^*$, so the optimal

$\alpha_N = \alpha_0 + \frac{n}{2}$. Finally, we have:

$$\begin{aligned}
\beta_N &= \beta^* \\
&= \frac{\alpha_N}{(n+\lambda_0+\lambda_N)(n+2\alpha_0)}\Big((n+\lambda_0+\lambda_N)(S+2\beta_0) + n\lambda_0(\bar{x}-\mu_0)^2 + n\lambda_N(\bar{x}-\mu_N)^2 \\
&+ \lambda_0\lambda_N(\mu_0-\mu_N)^2\Big) \\
&= \beta_0 + \frac{S}{2} + \frac{n\lambda_0(\bar{x}-\mu_0)^2 + n\lambda_N(\bar{x}-\mu_N)^2 + \lambda_0\lambda_N(\mu_0-\mu_N)^2}{2(n+\lambda_0+\lambda_N)} \\
&= \beta_0 + \frac{S}{2} + \frac{n\lambda_0(\bar{x}-\mu_0)^2}{2(n+\lambda_0)}
\end{aligned}$$

2.5.2 Gaussian approximation

For our general Gaussian approximation methods, the algorithm involves several steps as described below. These steps are summarized in Algorithm 2.2 and Algorithm 2.3.

Finding non-orthonormal basis of $T_{\theta_0}\Theta$ In this paper, we define the derivatives of the map $f : \mathbb{R}^n \to \mathbb{R}^m$ as $[\frac{d\mathbf{f}}{d\mathbf{x}}]_{ij} = \frac{\partial f_i(\mathbf{x})}{x_j}$ and we use the following notations [22], [54]:

$A \otimes B$**:** Kronecker product

$A \circ B$**:** Hadamard (elementwise) product

*vec***:** an operator which vectorizes the matrix column-wisely

*vech***:** an operator which vectorizes the matrix column-wisely but excludes the upper part of the matrix

$T_{m,n}$**:** $T_{m,n}vec(A_{m\times n}) = vec(A^T)$

R_n**:** $R_n vec(A_{n\times n}) = vec(A_{n\times n} \circ I)$

S_n: $vech(A_{n\times n}) = S_n vec(A_{n\times n})$

We calculate the basis as follows:

$$\begin{aligned}
v_\mu &= \frac{\partial q}{\partial \mu}\Big|_{\mu=\mu_0, L=L_0} \\
&= q(z|\mu_0, \Sigma_0)\frac{1}{2}(z-\mu_0)^T\Sigma_0^{-1}, \quad (1\times D \text{ vector}) \\
v_l &= \frac{\partial q}{\partial vech(L)}\Big|_{\mu=\mu_0, L=L_0} \\
&= q(z|\mu_0, \Sigma_0)\left[-\frac{1}{4}vec(\Sigma_0^{-T})^T + \frac{1}{4}((z-\mu_0)^T\otimes(z-\mu_0)^T)(\Sigma_0^{-T}\otimes\Sigma_0^{-1})\right] \\
&\left[I + T_{D,D}^T - R_D^T\right]\left[(I_D\otimes L_0)T_{D,D} + (L_0\otimes I_D)\right]S_D^T, \quad (1\times \frac{D(D+1)}{2} \text{ vector})
\end{aligned}$$

The basis v_l is obtained by using the chain rule:

$$\frac{\partial q}{\partial vech(L)} = \frac{\partial q}{\partial vec(\Sigma)}\frac{dvec(\Sigma)}{dvech(L)}$$

If we assume that Σ is an unstructured matrix (i.e. the entries of Σ are entirely independent), we have:

$$\frac{dq}{dvec(\Sigma)} = q(z|\mu,\Sigma)\left[-\frac{1}{4}vec(\Sigma^{-T})^T + \frac{1}{4}((z-\mu)^T\otimes(z-\mu)^T)(\Sigma^{-T}\otimes\Sigma^{-1})\right]$$

But because Σ is symmetric, the general rules do not apply. Instead, we have [1]

$$\frac{\partial q}{\partial \Sigma} = \frac{dq}{d\Sigma} + \left(\frac{dq}{d\Sigma}\right)^T - \frac{dq}{d\Sigma}\circ I$$

[1] We use symbol d for derivatives with respect to unstructured Σ and symbol ∂ for symmetric Σ

Correspondingly,

$$\begin{aligned}\frac{\partial q}{\partial vec(\Sigma)} &= vec\left(\frac{dq}{d\Sigma}\right)^T + vec\left(\frac{dq}{d\Sigma}^T\right)^T - vec\left(\frac{dq}{d\Sigma}\circ I\right)^T \\ &= \frac{dq}{dvec(\Sigma)} + \frac{dq}{dvec(\Sigma)}T_{D,D}^T - \frac{dq}{dvec(\Sigma)}R_D^T\end{aligned}$$

Finally, we have:

$$\begin{aligned}\frac{dvec(\Sigma)}{dvech(L)} &= \frac{dvec(LL^T)}{dvech(L)} \\ &= \left[(I_D\otimes L)T_{D,D} + (L\otimes I_D)\right]\frac{dvec(L)}{dvech(L)} \\ &= \left[(I_D\otimes L)T_{D,D} + (L\otimes I_D)\right]S_D^T\end{aligned}$$

Finding the inner products of the basis Notice that the inner product in $\mathcal{Q}$ is defined as:

$$\langle\varphi,\phi\rangle = \int_{-\infty}^{+\infty}\varphi\phi dz$$

We can then find the corresponding inner products,

$$\begin{aligned}\langle v_{\mu_i}, v_{\mu_j}\rangle &= \frac{1}{4}(\Sigma_0^{-1})_{ij} = \frac{1}{4}\sigma^{ij} \\ \langle v_{\mu_i}, v_{lj}\rangle &= 0 \\ \langle v_{li}, v_{lj}\rangle &= B_{ij}\end{aligned}$$

where B is the inner product matrix for v_l. We show how to obtain B. For simplicity, we denote $v_l = q(z|\mu_0, \Sigma_0)W_D U_D V_D$, where

$$\begin{aligned} W_D &= -\frac{1}{4}vec(\Sigma_0^{-T})^T + \frac{1}{4}((z-\mu_0)^T \otimes (z-\mu_0)^T)(\Sigma_0^{-T} \otimes \Sigma_0^{-1}) \\ U_D &= I + T_{D,D}^T - R_D^T \\ V_D &= \left[(I_D \otimes L_0)T_{D,D} + (L_0 \otimes I_D)\right] S_D^T \end{aligned}$$

Thus, $B = E_{q^2(z|\mu_0,\Sigma_0)}(V_D^T U_D^T W_D^T W_D U_D V_D) = V_D^T U_D^T E_{q^2(z|\mu_0,\Sigma_0)}(W_D^T W_D) U_D V_D$. We can calculate $E(W_D^T W_D)$ as follows:

$$\begin{aligned} E(W_D^T W_D) &= E(\left[-\frac{1}{4}vec(\Sigma_0^{-T}) + \frac{1}{4}(\Sigma_0^{-T} \otimes \Sigma_0^{-1})((z-\mu_0) \otimes (z-\mu_0))\right] \\ &\quad \left[-\frac{1}{4}vec(\Sigma_0^{-T})^T + \frac{1}{4}((z-\mu_0)^T \otimes (z-\mu_0)^T)(\Sigma_0^{-T} \otimes \Sigma_0^{-1})\right]) \\ &= \frac{1}{16}vec(\Sigma_0^{-1})vec(\Sigma_0^{-1})^T - \frac{1}{16}vec(\Sigma_0^{-1})E((z-\mu_0)^T \otimes (z-\mu_0)^T)(\Sigma_0^{-1} \otimes \Sigma_0^{-1}) \\ &\quad -\frac{1}{16}(\Sigma_0^{-T} \otimes \Sigma_0^{-1})E((z-\mu_0) \otimes (z-\mu_0))vec(\Sigma_0^{-1})^T \\ &\quad +\frac{1}{16}(\Sigma_0^{-1} \otimes \Sigma_0^{-1})E((z-\mu_0) \otimes (z-\mu_0))((z-\mu_0)^T \otimes (z-\mu_0)^T)(\Sigma_0^{-1} \otimes \Sigma_0^{-1}) \end{aligned}$$

Notice that we assume z follows a normal distribution, then according to Isserlis' theorem, we have:

$$\begin{aligned} E\left((z_i-\mu_i)(z_j-\mu_j)(z_k-\mu_k)\right) &= 0 \\ E\left((z_i-\mu_i)(z_j-\mu_j)(z_k-\mu_k)(z_l-\mu_l)\right) &= \sigma_{ij}\sigma_{kl} + \sigma_{ik}\sigma_{jl} + \sigma_{jk}\sigma_{il} \end{aligned}$$

Therefore,

$$\begin{aligned} E((z-\mu)^T \otimes (z-\mu)^T) &= vec(\Sigma^T)^T \\ E((z-\mu) \otimes (z-\mu)) &= vec(\Sigma^T) \\ E((z-\mu)(z-\mu)^T \otimes (z-\mu)(z-\mu)^T) &= [(\Sigma \otimes \Sigma)_{ijkl}] + [(\Sigma \otimes \Sigma)_{ikjl}] + [(\Sigma \otimes \Sigma)_{iljk}] \end{aligned}$$

$[(\Sigma \otimes \Sigma)_{ijkl}], [(\Sigma \otimes \Sigma)_{ikjl}], [(\Sigma \otimes \Sigma)_{iljk}]$ correspond to different permutations of $\Sigma \otimes \Sigma$ such that the matrix elements are $\sigma_{ij}\sigma_{kl}, \sigma_{ik}\sigma_{jl}, \sigma_{jk}\sigma_{il}$ respectively. Finally, we have

$$\begin{aligned} E(W_D^T W_D) = &\frac{1}{16} vec(\Sigma_0^{-1}) vec(\Sigma_0^{-1})^T - \frac{1}{16} vec(\Sigma_0^{-1}) vec(\Sigma_0)^T (\Sigma_0^{-1} \otimes \Sigma_0^{-1}) \\ &- \frac{1}{16} (\Sigma_0^{-T} \otimes \Sigma_0^{-1}) vec(\Sigma_0) vec(\Sigma_0^{-1})^T + \frac{1}{16} (\Sigma_0^{-1} \otimes \Sigma_0^{-1}) ([(\Sigma_0 \otimes \Sigma_0)_{ijkl}] \\ &+ [(\Sigma_0 \otimes \Sigma_0)_{ikjl}] + [(\Sigma_0 \otimes \Sigma_0)_{iljk}]) (\Sigma_0^{-1} \otimes \Sigma_0^{-1}) \end{aligned}$$

Orthonormalizing the basis Because $\langle v_{\mu_i}, v_{lj} \rangle = 0$, we only need to orthonormalize the set of basis $\{v_{\mu_i}\}_{i=1}^{D}$ and $\{v_{li}\}_{i=1}^{\frac{D(D+1)}{2}}$ respectively. We use the Gram-Schmidt process to find

the orthonormal basis. For $\{v_{\mu_i}\}_{i=1}^{D}$, for example, we have

$$\begin{aligned} w_{\mu_j} &= \frac{1}{\sqrt{D_{j-1}D_j}} \begin{vmatrix} \langle \mathbf{v}_{\mu_1}, \mathbf{v}_{\mu_1} \rangle & \langle \mathbf{v}_{\mu_2}, \mathbf{v}_{\mu_1} \rangle & \dots & \langle \mathbf{v}_{\mu_j}, \mathbf{v}_{\mu_1} \rangle \\ \langle \mathbf{v}_{\mu_1}, \mathbf{v}_{\mu_2} \rangle & \langle \mathbf{v}_{\mu_2}, \mathbf{v}_{\mu_2} \rangle & \dots & \langle \mathbf{v}_{\mu_j}, \mathbf{v}_{\mu_2} \rangle \\ \vdots & \vdots & \ddots & \vdots \\ \langle \mathbf{v}_{\mu_1}, \mathbf{v}_{\mu_{j-1}} \rangle & \langle \mathbf{v}_{\mu_2}, \mathbf{v}_{\mu_{j-1}} \rangle & \dots & \langle \mathbf{v}_{\mu_j}, \mathbf{v}_{\mu_{j-1}} \rangle \\ \mathbf{v}_{\mu_1} & \mathbf{v}_{\mu_2} & \dots & \mathbf{v}_{\mu_j} \end{vmatrix} \\ &= \frac{1}{\sqrt{D_{j-1}D_j}} \sum_{i=1}^{j} (-1)^{j+i} \mathbf{v}_{\mu_i} M_{j,i} \\ &= \frac{1}{\sqrt{D_{j-1}D_j}} \sum_{i=1}^{j} (-1)^{j+i} q(z|\mu_0, \Sigma_0) \frac{1}{2} \left[(z-\mu_0)^T \Sigma_0^{-1} \right]_i M_{j,i} \end{aligned}$$

where $D_0 = 1$ and D_j is the Gram determinant for $j \geqslant 1$:

$$D_j = \begin{vmatrix} \langle \mathbf{v}_{\mu_1}, \mathbf{v}_{\mu_1} \rangle & \langle \mathbf{v}_{\mu_2}, \mathbf{v}_{\mu_1} \rangle & \dots & \langle \mathbf{v}_{\mu_j}, \mathbf{v}_{\mu_1} \rangle \\ \langle \mathbf{v}_{\mu_1}, \mathbf{v}_{\mu_2} \rangle & \langle \mathbf{v}_{\mu_2}, \mathbf{v}_{\mu_2} \rangle & \dots & \langle \mathbf{v}_{\mu_j}, \mathbf{v}_{\mu_2} \rangle \\ \vdots & \vdots & \ddots & \vdots \\ \langle \mathbf{v}_{\mu_1}, \mathbf{v}_{\mu_j} \rangle & \langle \mathbf{v}_{\mu_2}, \mathbf{v}_{\mu_j} \rangle & \dots & \langle \mathbf{v}_{\mu_j}, \mathbf{v}_{\mu_j} \rangle \end{vmatrix}$$

$M_{j,i}$ is a minor of:

$$\begin{pmatrix} \langle \mathbf{v}_{\mu_1}, \mathbf{v}_{\mu_1} \rangle & \langle \mathbf{v}_{\mu_2}, \mathbf{v}_{\mu_1} \rangle & \dots & \langle \mathbf{v}_{\mu_j}, \mathbf{v}_{\mu_1} \rangle \\ \langle \mathbf{v}_{\mu_1}, \mathbf{v}_{\mu_2} \rangle & \langle \mathbf{v}_{\mu_2}, \mathbf{v}_{\mu_2} \rangle & \dots & \langle \mathbf{v}_{\mu_j}, \mathbf{v}_{\mu_2} \rangle \\ \vdots & \vdots & \ddots & \vdots \\ \langle \mathbf{v}_{\mu_1}, \mathbf{v}_{\mu_{j-1}} \rangle & \langle \mathbf{v}_{\mu_2}, \mathbf{v}_{\mu_{j-1}} \rangle & \dots & \langle \mathbf{v}_{\mu_j}, \mathbf{v}_{\mu_{j-1}} \rangle \\ \mathbf{v}_{\mu_1} & \mathbf{v}_{\mu_2} & \dots & \mathbf{v}_{\mu_j} \end{pmatrix}$$

, obtained by taking the determinant of this matrix with row j and column i removed. To simplify the calculation, we could equivalently treat $M_{j,i}$ as a minor of:

$$A_j = \begin{pmatrix} \langle \mathbf{v}_{\mu_1}, \mathbf{v}_{\mu_1} \rangle & \langle \mathbf{v}_{\mu_2}, \mathbf{v}_{\mu_1} \rangle & \dots & \langle \mathbf{v}_{\mu_j}, \mathbf{v}_{\mu_1} \rangle \\ \langle \mathbf{v}_{\mu_1}, \mathbf{v}_{\mu_2} \rangle & \langle \mathbf{v}_{\mu_2}, \mathbf{v}_{\mu_2} \rangle & \dots & \langle \mathbf{v}_{\mu_j}, \mathbf{v}_{\mu_2} \rangle \\ \vdots & \vdots & \ddots & \vdots \\ \langle \mathbf{v}_{\mu_1}, \mathbf{v}_{\mu_j} \rangle & \langle \mathbf{v}_{\mu_2}, \mathbf{v}_{\mu_j} \rangle & \dots & \langle \mathbf{v}_{\mu_j}, \mathbf{v}_{\mu_j} \rangle \end{pmatrix}$$

since row j is crossed out anyway. Notice that A_j is the jth order leading principal submatrix of $\frac{1}{4}\Sigma_0^{-1}$.

For basis $\{v_{li}\}_{i=1}^{\frac{D(D+1)}{2}}$, we have already obtained its Gram matrix B. We denote B_j the jth order leading principal submatrix of B. The rest of the procedure is similar to deriving $\{w_{\mu_i}\}_{i=1}^{D}$ from $\{v_{\mu_i}\}_{i=1}^{D}$.

Updating the approximating distribution We have

$$\overrightarrow{v_0} = \frac{\sum_{i=1}^{D} \langle w_{\mu_i}, \sqrt{p_0} \rangle w_{\mu_i} + \sum_{i=1}^{D(D+1)/2} \langle w_{li}, \sqrt{p_0} \rangle w_{li}}{\sqrt{1 - \langle \theta_0, \sqrt{p_0} \rangle^2}}$$

Therefore, we use the following updates:

$$\begin{aligned} \mu_i^{(t+1)} &= \mu_i^{(t)} + \epsilon_{\alpha_i} \langle w_{\mu_i}, \sqrt{p_0} \rangle / \sqrt{1 - \langle \theta_0, \sqrt{p_0} \rangle^2} \quad i = 1, \cdots, D \\ l_i^{(t+1)} &= l_i^{(t)} + \epsilon_{\beta_i} \langle w_{li}, \sqrt{p_0} \rangle / \sqrt{1 - \langle \theta_0, \sqrt{p_0} \rangle^2} \quad i = 1, \cdots, \frac{D(D+1)}{2} \end{aligned}$$

where $\epsilon_{\alpha_i}, \epsilon_{\beta_i}$ are stepsizes. As we can see, in order to update the parameters, all we need to calculate is $\langle w_{\mu_j}, \sqrt{p_0}\rangle$, $\langle w_{li}, \sqrt{p_0}\rangle$ and $\langle \theta_0, \sqrt{p_0}\rangle$:

$$\begin{aligned}
\langle w_{\mu_j}, \sqrt{p_0}\rangle &= \int \sqrt{p_0(z)} \frac{1}{\sqrt{D_{j-1}D_j}} \sum_{i=1}^{j} (-1)^{j+i} q(z|\mu_0, \Sigma_0) \frac{1}{2} \left[(z-\mu_0)^T \Sigma_0^{-1}\right]_i M_{j,i} dz \\
&= \frac{1}{\sqrt{D_{j-1}D_j}} \sum_{i=1}^{j} (-1)^{j+i} M_{j,i} E_{p(z|\mu_0,\Sigma_0)} \left(\frac{\sqrt{p_0(z)}}{q(z|\mu_0,\Sigma_0)} \frac{1}{2} \left[(z-\mu_0)^T \Sigma_0^{-1}\right]_i \right) \\
\langle w_{li}, \sqrt{p_0}\rangle &= \frac{1}{\sqrt{E_{j-1}E_j}} \sum_{i=1}^{j} (-1)^{j+i} N_{j,i} E_{p(z|\mu_0,\Sigma_0)} \left(\frac{\sqrt{p_0(z)}}{q(z|\mu_0,\Sigma_0)} \left[W_D U_D V_D\right]_i \right) \\
\langle \theta_0, \sqrt{p_0}\rangle &= E_{p(z|\mu_0,\Sigma_0)} \left(\frac{\sqrt{p_0(z)}}{q(z|\mu_0,\Sigma_0)} \right)
\end{aligned}$$

Here, E_j is the determinant of B_j and $N_{j,i}$ is the minor of B_j. Expectations with respect to $p(z|\mu_0, \Sigma_0)$ can be approximated by the Monte Carlo method.

2.5.3 Illustrative example: t-distribution

We now discuss the details for approximating $t(1)$ with a normal distribution. For a specific θ_0, the basis of the tangent space $T_{\theta_0}\Theta$ is

$$\begin{aligned}
v_\mu = \left.\frac{\partial q}{\partial \mu}\right|_{\mu=\mu_0,\sigma=\sigma_0} &= q(x|\mu_0, \sigma_0^2) \frac{1}{2\sigma_0^2}(x-\mu_0) \\
v_\sigma = \left.\frac{\partial q}{\partial \sigma}\right|_{\mu=\mu_0,\sigma=\sigma_0} &= q(x|\mu_0, \sigma_0^2) \left[-\frac{1}{2}(\sigma_0)^{-1} + \frac{1}{2}\sigma_0^{-3}(x-\mu_0)^2\right] \text{, where } -\infty < \sigma < \infty
\end{aligned}$$

with the corresponding inner product

$$
\begin{aligned}
\langle v_\mu, v_\mu \rangle &= \int_{-\infty}^{+\infty} v_\mu v_\mu dx \\
&= E_{q^2(x|\mu_0,\sigma_0^2)}(\frac{1}{2\sigma_0^2}(x-\mu_0)\frac{1}{2\sigma_0^2}(x-\mu_0)) \\
&= \frac{1}{4\sigma_0^2} \\
\langle v_\sigma, v_\sigma \rangle &= \int_{-\infty}^{+\infty} v_\sigma v_\sigma dx \\
&= E_{q^2(x|\mu_0,\sigma_0^2)}(\left[-\frac{1}{2}(\sigma_0)^{-1} + \frac{1}{2}\sigma_0^{-3}(x-\mu_0)^2\right]\left[-\frac{1}{2}(\sigma_0)^{-1} + \frac{1}{2}\sigma_0^{-3}(x-\mu_0)^2\right]) \\
&= \frac{1}{2}\sigma_0^{-2} \\
\langle v_\mu, v_\sigma \rangle &= \int_{-\infty}^{+\infty} v_\mu v_\sigma dx \\
&= E_{q^2(x|\mu_0,\sigma_0^2)}(\frac{1}{2\sigma_0^2}(x-\mu_0)\left[-\frac{1}{2}(\sigma_0)^{-1} + \frac{1}{2}\sigma_0^{-3}(x-\mu_0)^2\right]) = 0
\end{aligned}
$$

Therefore, we can obtain an orthonormal basis as follows:

$$
\begin{aligned}
w_\mu &= \frac{v_\mu}{\|v_\mu\|} = \frac{v_\mu}{\sqrt{\langle v_\mu, v_\mu \rangle}} \\
&= q(x|\mu_0,\sigma_0^2)\frac{1}{\sqrt{\sigma_0^2}}(x-\mu_0) \\
w_\sigma &= \frac{v_\sigma}{\|v_\sigma\|} = \frac{v_\sigma}{\sqrt{\langle v_\sigma, v_\sigma \rangle}} \\
&= q(x|\mu_0,\sigma_0^2)\frac{\sqrt{2}}{2}(\frac{(x-\mu_0)^2}{\sigma_0^2} - 1)
\end{aligned}
$$

Given $\overrightarrow{v_0} = \frac{\langle w_\mu, \sqrt{p_0}\rangle w_\mu + \langle w_\sigma, \sqrt{p_0}\rangle w_\sigma}{\sqrt{1 - \langle \theta_0, \sqrt{p_0}\rangle^2}}$, we use the following updates:

$$\begin{aligned}
\mu^{(t+1)} &= \mu^{(t)} + \epsilon_\alpha \frac{\langle w_\mu, \sqrt{p_0}\rangle}{\sqrt{1 - \langle \theta_0, \sqrt{p_0}\rangle^2}} \\
\sigma^{(t+1)} &= \sigma^{(t)} + \epsilon_\beta \frac{\langle w_\sigma, \sqrt{p_0}\rangle}{\sqrt{1 - \langle \theta_0, \sqrt{p_0}\rangle^2}}
\end{aligned}$$

We calculate $\langle w_\mu, \sqrt{p_0}\rangle, \langle w_\sigma, \sqrt{p_0}\rangle, \langle \theta_0, \sqrt{p_0}\rangle$ as follows

$$\begin{aligned}
\langle w_\mu, \sqrt{p_0}\rangle &= \int_{-\infty}^{+\infty} w_\mu \sqrt{p_0} dx \\
&= E_{q^2(x|\mu_0,\sigma_0^2)}\left(\frac{\pi^{-\frac{1}{2}}(1+x^2)^{-\frac{1}{2}}}{q(x|\mu_0,\sigma_0^2)} \frac{1}{\sqrt{\sigma_0^2}}(x-\mu_0)\right) \\
&= \left(\frac{\pi}{2\sigma_0^2}\right)^{-\frac{1}{4}} \frac{c_2}{\sqrt{\sigma_0^2}} \\
\langle w_\sigma, \sqrt{p_0}\rangle &= \int_{-\infty}^{+\infty} w_\sigma \sqrt{p_0} dx \\
&= E_{q^2(x|\mu_0,\sigma_0^2)}\left(\frac{\pi^{-\frac{1}{2}}(1+x^2)^{-\frac{1}{2}}}{q(x|\mu_0,\sigma_0^2)} \frac{\sqrt{2}}{2}\left(\frac{(x-\mu_0)^2}{\sigma_0^2} - 1\right)\right) \\
&= \left(\frac{\pi}{2\sigma_0^2}\right)^{-\frac{1}{4}} \left(\frac{\sqrt{2}c_1}{2\sigma_0^2} - \frac{\sqrt{2}c_3}{2}\right) \\
\langle \theta_0, \sqrt{p_0}\rangle &= \int_{-\infty}^{+\infty} q(x|\mu_0,\sigma_0^2)\sqrt{p_0} dx \\
&= E_{q^2(x|\mu_0,\sigma_0^2)}\left(\frac{\pi^{-\frac{1}{2}}(1+x^2)^{-\frac{1}{2}}}{q(x|\mu_0,\sigma_0^2)}\right) \\
&= \left(\frac{\pi}{2\sigma_0^2}\right)^{-\frac{1}{4}} c_3
\end{aligned}$$

where

$$
\begin{aligned}
c_1 &= E_{q_0^2}\left(\frac{(x-\mu_0)^2}{\sqrt{1+x^2}\exp(-\frac{1}{4\sigma_0^2}(x-\mu_0)^2)}\right) \\
c_2 &= E_{q_0^2}\left(\frac{x-\mu_0}{\sqrt{1+x^2}\exp(-\frac{1}{4\sigma_0^2}(x-\mu_0)^2)}\right) \\
c_3 &= E_{q_0^2}\left(\frac{1}{\sqrt{1+x^2}\exp(-\frac{1}{4\sigma_0^2}(x-\mu_0)^2)}\right.
\end{aligned}
$$

Chapter 3

Speeding Hamiltonian Monte Carlo with Auto-encoder

In this paper, we propose a new Hamiltonian Monte Carlo (HMC) sampling scheme based on auto-encoders. We find a low dimensional representation of the parameter space and perform HMC in the latent space, and the new state is projected back to the original space as a Metropolis proposal. While the induced dynamics in the parameter space is no longer Hamiltonian, it's still time reversible, and the Markov chain still converges to the canonical distribution with a volume correction term. We evaluated our method on high-dimensional simulated data and text classification tasks. The empirical results show that our method maintains a competitive ability to explore high-dimensional space with complicated geometry, and obtains an decent prediction accuracy while achieving great computational savings compared to standard HMC.

3.1 Introduction

Recent years' advances in computational resource have greatly contributed to the development and popularity of Bayesian statistics. However, many problems in today's world have very high-dimensional features, from text and image processing to scientific applications such as neuron science and astronomy. To handle the high-dimension and complicated structures of the data, more advanced MCMC techniques such as Langevin dynamics and Hamiltonian Monte Carlo (HMC) ([53]) have been developed.

In particular, HMC is efficient in exploring the state space by exploiting the gradient of the target density, which reflects local geometric information of the probability distribution, but at the cost of more computational resources. Endeavors have been put to find a balance between efficiency and computational savings for HMC ([60], [65], [64], 41).

HMC sampling in high dimensional space is even more computationally intensive. However, in many cases, the feature space or parameter space could be redundant. An efficient exploration of its latent space might be sufficient for good results in the original space, while exploration in a lower dimensional space could be less computationally intensive. While this could be achieved by performing feature extraction first, we propose Auto-encoding HMC, which handles this situation naturally by performing feature extraction when sampling in the original parameter space.

We organize our paper as follows. We first reviewed the difficulty of MCMC exploration in the high dimensional space (mostly due to its complicated geometry), and why HMC can deal with this challenge. We then described our method, auto-encoding HMC, in details. We also proved that our sampling method will generate a Markov chain which converges to the target distribution. Finally, we justified our method by providing empirical results for a simulated data analysis and a real world application using Bayesian logistic regression model, and compared our results with standard HMC.

3.2 Preliminaries

One central task of Bayesian inference is to calculate the high-dimensional integral:

$$E_\pi(f) = \int \pi(q) f(q) dq$$

, where $\pi(q) = p(q)p(D|q)$ is the posterior distribution with respect to parameter q. The integral is not analytically solvable in general, and we usually resort to numerical sampling to obtain samples from $\pi(q)$. The samples are used to calculate a finite sum for approximating the integral.

An accurate approximation usually relies on efficient exploration of typical set, the region in the parameter space which contributes most to the integral ([10]).

3.2.1 Pathological Behavior of Random Walk Metropolis in High dimensional space

One most well-known sampling method is Markov chain Monte Carlo (MCMC). MCMC method samples from the space by generating a Markov chain which eventually converges to the target distribution (the posterior distribution in Bayesian framework) as its stationary distribution. A new state is proposed at each iteration according to a transition map $T(q^*|q)$.

In particular, Metropolis-Hastings algorithm is used to construct such a Markov chain by proposing a new state and accept it with the following probability:

$$a(q, q^*) = \min(1, \frac{\pi(q^*)T(q|q^*)}{\pi(q)T(q^*|q)})$$

, which can guarantee the convergence to $\pi(q)$ due to detailed balance. Random walk

Metropolis is one of the most widely used Metropolis algorithm, with $T(q^*|q)$ to be a Gaussian distribution centered on current state q.

Though random walk Metropolis is simple to implement, it does not scale well to the dimension of the parameter space — exploration of typical set in high dimensional space is very challenging for random walk proposals. As [10] explained, the region outside the typical set has vanishing densities and large volume, which does not contribute a lot to the integral. Thus it does not worth exploring this area with too much computational resource. As the dimension of the space grows, the volume of the outside region grows exponentially, and overwhelms the volume interior. Thus random walk Metropolis will always propose a state outside the typical set and get rejected.

3.2.2 Hamiltonian Monte Carlo

Instead of randomly jumping around the typical set, a MCMC algorithm which could efficiently explore the typical set in high dimensional space is apparently very attractive. Hamiltonian Monte Carlo is such a MCMC method which exploits the geometric information (gradient) of the probability distribution, and thus scales well to high-dimensional problems and those with complicated geometry.

In particular, HMC introduces a set of auxiliary variables named momentum p. Each p_i corresponds to each dimension of the parameter q_i, where $i = 1, \cdots, D$. The parameter space is then expanded to a phase space (q, p), and HMC proposes new states jointly for (q, p), according to Hamilton's equations:

$$\frac{dq_i}{dt} = \frac{\partial H}{\partial p_i}$$
$$\frac{dp_i}{dt} = -\frac{\partial H}{\partial q_i}$$

where $H = H(q,p) = U(q) + K(p)$. $U(q)$ is associated with the target density, and $K(p)$ is usually chosen to associate with the density of zero-mean Gaussian with covariance M.

$$U(q) = -\log \pi(q) = -\log[p(q)p(D|q)]$$
$$K(p) = p^T M^{-1} p/2$$

A new state will be proposed from $(q(t), p(t))$ to $(q(t+s), p(t+s))$. Since the Hamiltonian equations are not analytically solvable in general, in practice, we resort to leapfrog method by discretizing the time to approximate the dynamics. Given a step size ϵ and number of steps L, s is defined to be ϵL.

Hamilton's equations describes the dynamics of a physical system with conservative energy $H(q,p)$. q is the position of the object, and p is the momentum. Correspondingly, $U(q)$ is the potential energy, and $K(p)$ is the kinetic energy. While $U(q)$ and $K(p)$ are varying as the object moves, the Hamiltonian $H(q,p) = U(q) + K(p)$ is conservative.

In its MCMC application, $\exp(-H(q,p))$ corresponds to the joint probability of (q,p), also referred to as canonical distribution:

$$\pi(q,p) = \frac{1}{Z} \exp(-H(q,p)/T)$$

where Z is a normalization constant, and T represents the temperature of the system. Eventually the Markov chain will converge to the canonical distribution due to the reversibility and volume preservation properties of the Hamiltonian dynamics. The marginal distribution of q is exactly the target density.

3.2.3 Auto-encoder

Auto-encoder is a special type of feed forward neural network for learning latent representation of the data (Figure 3.1). The data is fed from the input layer and encoded into a low-dimensional latent representation (code). The code is then decoded into a reconstruction of the original data. The goal of auto-encoder is to learn an identity map such that the output (reconstruction) is closely matched with the input data. The model is trained to minimize the difference between the input and the reconstruction. Auto-encoder could learn complicated nonlinear dimensionality reduction and thus is widely used in challenging tasks such as image recognition and artificial data generation.

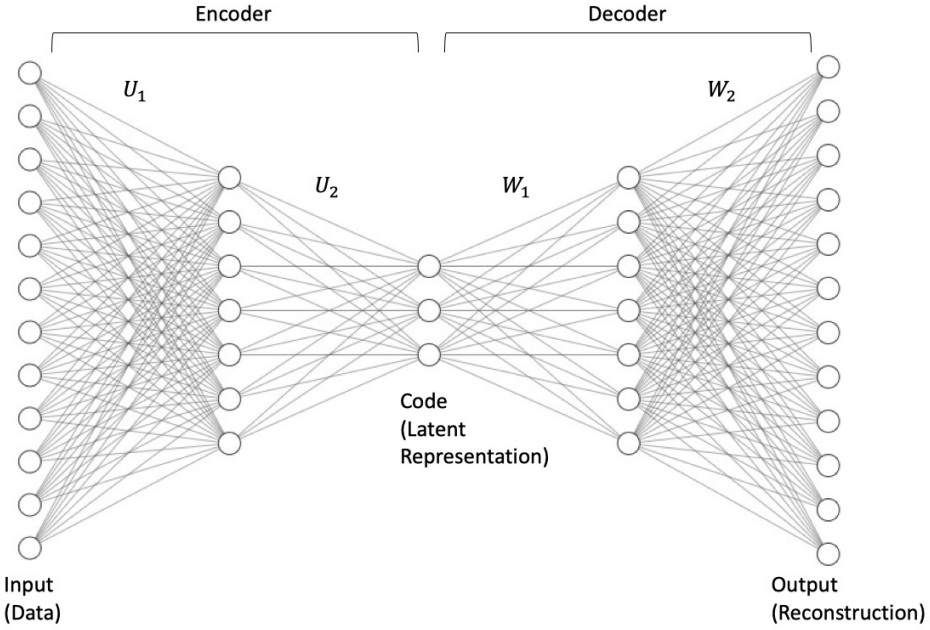


Figure 3.1: Auto-encoder Network Architecture

According to universal approximation theorem ([17]), a feed-forward artificial neural network can approximate any continuous function given some mild assumptions about the activation functions. Theoretically, an auto-encoder with suitable activation functions could represent an identity map. Therefore, auto-encoder could learn a encoder ϕ and decoder ψ such that $\phi \circ \psi = I$. An accurate reconstruction of the data implies a good low-dimensional

representation.

3.3 Auto-encoding HMC

One disadvantage of HMC is that single iteration is more computationally demanding since we have to evaluate the gradient information. We thus propose Auto-encoding HMC, in the effort of reducing computational costs, while still maintaining the efficiency of HMC.

We propose to first collect a small set of posterior samples from the target density by running standard HMC. We then perform dimensionality reduction for the collected samples to find a latent space of the parameter space. We then simulate Hamiltonian dynamics in the latent space, and project the new state back to the original space to obtain proposals. While this exploration will not be as accurate as the standard HMC, it could reduce a certain amount of computational costs .

We now illustrate the computational benefits with a three-dimensional Gaussian example. Notice that the application of our method is only for high-dimensional problems. Because auto-encoder will over-fit a low-dimensional problem, we instead use Principle Component Analysis (PCA) here for dimensionality reduction. It can be shown that the encoder of an auto-encoder reduces to a PCA if all the activation functions are linear and the inputs are normalized.

Suppose we are interested in sampling from a three-dimensional Gaussian distribution with zero mean and covariance

$$\Sigma = \begin{bmatrix} 1 & 0.95 & 0.7 \\ 0.95 & 1 & 0.5 \\ 0.7 & 0.5 & 1 \end{bmatrix}$$

. To perform dimensionality reduction using PCA, we simply find orthornormal matrix

P such that $\Sigma' = P\Sigma P^T$ is diagonalized, where Σ' is the covariance of the transformed variables. Here we extract the first two principal components with greatest variance as the low-dimensional representation, and simulate Hamiltonian dynamics in the latent space.

As shown in Figure 3.2, 3.3, we only performed HMC in the space of two dimensions. But when the proposals are projected back to the original space, the algorithm still efficiently explores the space with distant proposals.

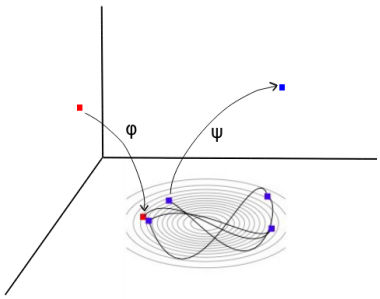


Figure 3.2: HMC trajectory in the latent space (2-dimensional), with the red square to be the initialized position, and blue squares HMC proposals.

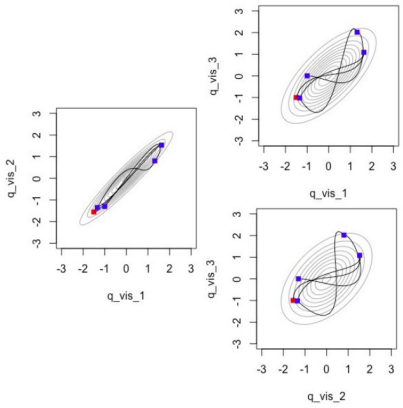


Figure 3.3: Trajectories projected back to parameter space (3-dimensional)

When it comes to high-dimensional problems, we will use auto-encoder for dimensionality reduction. More specifically, let's denote the parameters of interest q_v and its latent representation q_h. We also denote the encoder and decoder:

$$\phi : q_v \mapsto q_h$$

$$\psi : q_h \mapsto q'_v$$

, where q'_v is a reconstruction of q_v. If the error of the auto-encoder goes to zero, we have:

$$\psi = \phi^{-1} : q_h \mapsto q_v$$

Our algorithm is composed of the following three stages:

1. Pre-sample a few (e.g. 1000) samples of q_v using standard HMC
2. Train an auto-encoder to fit the samples, and obtain fitted encoder ϕ and decoder ψ
3. Run Auto-encoding HMC to propose q_v^* from q_v (a detailed version in Algorithm 3.4):

   i Calculate $q_h = \phi(q_v)$

   ii Propose q_h^* from q_h by running HMC in the latent space

   iii Obtain $q_v^* = \psi(q_h^*)$

3.3.1 HMC in the latent space

Let's denote the complementary momentum of q_v to be p_v. The corresponding latent space parameters are denoted (q_h, p_h). The auxiliary variables in the latent space is constructed using the same learned encoder $p_h = \phi(p_v)$, with p_v sampled from a Gaussian distribution.

Let's denote the target density $\pi_{q_v}(q_v)$. We choose the potential energy of the latent space to be the negative log of $\pi_{q_v}(q_v)$:

$$U_h(q_h) = -\log \pi_{q_v}(\psi(q_h))$$

Notice that this is not a re-parameterization such that we will evaluate the integral using a new probability density. Instead, we directly use the potential energy function from the original space. If we use the density function of q_h induced by $\phi(q_v)$ to be the potential

energy, we will need to evaluate the volume change at each leapfrog step, which increases computational costs. As long as we could ensure detailed balance in the original space, which we will prove in later section, the MCMC proposal mechanism will be valid.

We also set the kinetic energy

$$K_h(p_h) = K_v(p_v) = p_v^T M^{-1} p_v/2 \tag{3.1}$$

Thus we simulate the following Hamiltonian Monte Carlo in the latent space:

$$\frac{dq_{hi}}{dt} = \frac{\partial K_h(p_h)}{\partial p_{hi}}$$
$$\frac{dp_{hi}}{dt} = -\frac{\partial U_h(q_h)}{\partial q_{hi}}$$

The evaluation of the gradient of the potential function with respect to q_h can be calculated by chain rules. For example, in our experiments, the decoder has one hidden layer with activation function $tanh$, and the output layer to be linearly connected. We can calculate the gradient function with respect to the latent q_h as follows :

$$\begin{aligned} \frac{\partial U_h(q_h)}{\partial q_h} &= \frac{\partial U_h(q_h)}{\partial q_v} \cdot \frac{\partial q_v}{\partial q_h} \\ \text{where} \quad \frac{\partial q_v}{\partial q_h} &= W_2 \cdot diag(1 - \tanh^2(W_1 q_h + b_1)) \cdot W_1 \\ \tanh(z) &= \frac{e^z - e^{-z}}{e^z + e^{-z}}, \quad \tanh'(z) = 1 - \tanh^2(z) \end{aligned} \tag{3.2}$$

where W_1 and W_2 are the learned weights of the decoder (Figure 3.1). A detailed calculation of the gradient of $U(q_h)$ regarding a logistic regression example can be found in the Appendix 3.6.1. The resulting gradient evaluation is less expensive because of the much lower dimension.

The evaluation of $\frac{\partial K_h(p_h)}{\partial p_h}$ is similar. With $\frac{\partial K_h(p_h)}{\partial p_v} = M^{-1} p_v$ and $p_v = W_2 \tanh(W_1 p_h + b_1)$,

we have:

$$\begin{aligned}\frac{\partial K_h(p_h)}{\partial p_h} &= \{M^{-1}W_2 \tanh(W_1 p_h + b_1)\}^T W_2 \cdot diag(1 - \tanh^2(W_1 p_h + b_1)) \cdot W_1 \\ &= \tanh(W_1 p_h + b_1)^T W_2^T M^{-1} W_2 \cdot diag(1 - \tanh^2(W_1 p_h + b_1)) \cdot W_1 \end{aligned} \tag{3.3}$$

where $W_2^T M^{-1} W_2$ can be pre-calculated.

In practice, the Hamiltonian dynamics is simulated using leapfrog method.

3.3.2 Proposal and correction

Joint distribution of (q_v, p_v) The density of p_v is selected to be zero-mean Gaussian with a covariance M, corresponding to $K_v(p_v)$ defined in equation (3.1). We then have the canonical distribution of the original phase space:

$$\pi_{q_v,p_v}(q_v, p_v) \propto \exp(\log \pi_{q_v}(q_v) - p_v^T M^{-1} p_v/2) \tag{3.4}$$

Notice the induced dynamics in the phase space (q_v, p_v) is no longer Hamiltonian, and does not have the property of volume preservation as standard HMC. We hereby prove that the proposed HMC update will leave the canonical distribution for q_v and p_v (equation 3.4) invariant, if we have that:

i the update is time reversible and thus symmetrical

ii an appropriate volume correction term is added in the HMC acceptance probability

Time reversibility The proof is straightforward. Let T_s represent the Hamiltonian dynamic in the latent space from the state (q_h, p_h) at time t to the state (q_h^*, p_h^*) at time $t + s$. The

reversibility of Hamiltonian dynamics indicates that:

$$T_s(q(t), p(t)) = (q(t+s), p(t+s))$$
$$T_s(q(t+s), -p(t+s)) = (q(t), -p(t))$$

If we let $Q(q_h^*, p_h^*|q_h, p_h)$ represent the process of negating momentum, applying mapping T_s and negating the momentum again, and let $\Phi(q_v, p_v) = (\phi(q_v), \phi(p_v))$, $\Psi(q_h, p_h) = (\psi(q_h), \psi(p_h))$, our proposal can be denoted $Q' = \Psi \circ Q \circ \Phi$. We must have:

$$Q'(q_v^*, p_v^*|q_v, p_v) = Q'(q_v, p_v|q_v^*, p_v^*)$$

A detailed proof can be found in Appendix 3.6.2.

Detailed Balance with Volume Correction Following the proof in [53], let's show that when accounting for volume change in the acceptance ratio, detailed balance holds for our proposed Metropolis update.

Consider partitioning the phase space (q, p) into small regions A_k with small volume V. Suppose by applying mapping Q' to A_k, the image of A_k becomes B_k. The B_k will also partition the space due to reversibility, but has a different volume V'. We need to show detailed balance:

$$P(A_i)T(B_j|A_i) = P(B_j)T(A_i|B_j) \quad \forall i, j$$

Since when $i \neq j$, $T(B_j|A_i) = T(A_i|B_j) = 0$, we only consider when $i = j \equiv k$. Let

$$T(B_k|A_k) = Q'(B_k|A_k) \min(1, \frac{\exp(-H_{B_k})}{\exp(-H_{A_k})} \frac{V'}{V})$$

We then have

$$\begin{aligned}
P(A_k)T(B_k|A_k) &= V\exp(-H_{A_k})Q'(B_k|A_k)\min(1,\frac{\exp(-H_{B_k})}{\exp(-H_{A_k})}\frac{V'}{V}) \\
&= Q'(B_k|A_k)\min(V\exp(-H_{A_k}), V'\exp(-H_{B_k})) \\
&= Q'(A_k|B_k)\min(V\exp(-H_{A_k}), V'\exp(-H_{B_k})) \\
&= V'\exp(-H_{B_k})Q'(B_k|A_k)\min(\frac{\exp(-H_{A_k})}{\exp(-H_{B_k})}\frac{V}{V'}, 1) \\
&= P(B_k)T(A_k|B_k)
\end{aligned}$$

The volume correction term $\frac{V'}{V}$ is simply the determinant of the Jacobian matrix $\left|\frac{\partial(q_v^*, p_v^*)}{\partial(q_v, p_v)}\right|$.

Calculation of acceptance ratio For acceptance ratio $\alpha = \min(1, \rho)$, we have

$$\begin{aligned}
\rho &= \exp(-H(q_v^*, p_v^*) + H(q_v, p_v))\left|\frac{\partial(q_v^*, p_v^*)}{\partial(q_v, p_v)}\right| \\
&= \exp(-U_v(q_v^*) + U_v(q_v) - K_v(p_v^*) + K_v(p_v))\left|\frac{\partial(q_v^*, p_v^*)}{\partial(q_v, p_v)}\right|
\end{aligned}$$

The determinant of $\frac{\partial(q_v^*, p_v^*)}{\partial(q_v, p_v)}$ is infeasible to evaluate. We showed in Appendix 3.6.3 that $\left|\frac{\partial(q_v^*, p_v^*)}{\partial(q_v, p_v)}\right|$ can be approximated by $\frac{Vol(q_v^*, p_v^*)}{Vol(q_h^*, p_h^*)}\frac{Vol(q_h, p_h)}{Vol(q_v, p_v)}$. These two Jacobian matrices are not full rank, so we use the square root of its gramian function $G(\cdot)$ to calculate the volume change (Appendix 3.6.3). Thus we have:

$$\rho = \exp(-U_v(q_v^*) + U_v(q_v) - K_v(p_v^*) + K_v(p_v))\sqrt{G(\frac{\partial(q_v^*, p_v^*)}{\partial(q_h^*, p_h^*)})}\sqrt{G(\frac{\partial(q_h, p_h)}{\partial(q_v, p_v)})} \tag{3.5}$$

Algorithm 3.4 summarizes the steps for a single iteration of Auto-encoding HMC.

Algorithm 3.4 Auto-encoding HMC

Inputs:
encoder ϕ, decoder ψ
$U_v(q_v)$
$grad\_U_h(q_h)$ according to equation (3.2)
$grad\_K_h(p_h)$ according to equation (3.3)
auto-encoder weights and biases W, b
step size ϵ, number of leapfrog steps L
current q_v

Initialize $q_v^{(0)} =$ current q_v
Sample momentum $p_v^{(0)} \sim \text{Normal}(0, M)$
Calculate $q_h^{(0)} = \phi(q_v^{(0)})$
Calculate $p_h^{(0)} = \phi(p_v^{(0)})$
for $i = 1$ to L **do**
$p_h^{(i-1/2)} = p_h^{(i-1)} - \epsilon/2 \cdot grad\_U_h(q_h^{(i-1)})$
$q_h^{(i)} = q_h^{(i-1)} + \epsilon \cdot grad\_K_h(p_h^{(i-1/2)})$
$p_h^{(i)} = p_h^{(i-1/2)} - \epsilon/2 \cdot grad\_U_h(q_h^{(i)})$
end for
Calculate $q_v^{(L)} = \psi(q_h^{(L)})$
Calculate $\rho = \exp(-H(q_v^{(L)}, p_v^{(L)}) + H(q_v^{(0)}, p_v^{(0)})) \cdot \left| \dfrac{\partial(q_v^{(L)}, p_v^{(L)})}{\partial(q_v^{(0)}, p_v^{(0)})} \right|$ according to equation (3.5)
(W, b will be needed accordingly)
Sample $u \sim \text{Uniform}(0, 1)$
if $u < \min(1, \rho)$ **then**
return $q_v^* = q_v^{(L)}$
else
return $q_v^* =$ current q_v
end if

3.4 Experiments

3.4.1 Implementation Details

Architecture of Auto-encoder The auto-encoder in our experiments is a five-layer neural network, with one input layer, one output layer and three hidden layers. The second hidden layer is the code, i.e., the latent representation. We used tanh as the activation function except for the linear output layer.

Let's denote the weights for each layer to be U_1, U_2, W_1, W_2 as shown in Figure 3.1, and d_1, b_1 to be biases corresponding to U_1, W_1. For this particular architecture, we have

$$\frac{\partial(q_v^*)}{\partial(q_h^*)} = W_2 \cdot diag(1 - \tanh^2(W_1 q_h^* + b_1)) \cdot W_1$$
$$\frac{\partial(q_h)}{\partial(q_v)} = diag(1 - q_h^2) \cdot U_2 \cdot diag(1 - \tanh^2(U_1 q_v + d_1)) \cdot U_1$$

3.4.2 High-dimensional Bayesian logistic regression with simulated data

To demonstrate that Auto-encoding HMC maintains the ability to explore the parameter space while greatly reducing the computational time, we conducted a high dimensional logistic regression experiment with simulated data.

Settings In this experiment, the dimension of the parameter space is 500, and we select the latent dimension to be 50. We also select the dimension of other hidden layers to be 100. Design matrix X is independently sampled from $N(0, 0.1^2)$. True β is uniformly sample from $[-1, 1]$. And we set its prior distribution to be $N(0, 10^2)$.

We first run standard HMC, and the first 1000 accepted proposals were used to train the auto-encoder (acceptance rate is set approximately 1). We normalized the input for better training results, and adjusted some calculation terms accordingly in the auto-encoding HMC proposal.

For both standard HMC and Auto-encoding HMC, we tune the acceptance rate to be around 0.65 to 0.7, which is optimal in terms of computational efficiency ([8]). We compared the results using 1000 samples after convergence has been reached.

Results The computational time of auto-encoding HMC is 0.134 second per iteration, but it costs 0.488 second for standard HMC. The minimum effective sample size per second (minESS/s) is 0.964 for our method, and 0.203 for standard HMC.

Our method is faster with higher minESS/s, though at the comprise of exploring less regions compared to standard HMC. As shown in Figure 3.4, where we visualize the posterior distributions of β's for 5 dimensions for both sampling methods. Our method still provides relatively efficient proposals, which is a good approximation to the samples obtained by standard HMC. A quick check for other dimensions shows similar results.

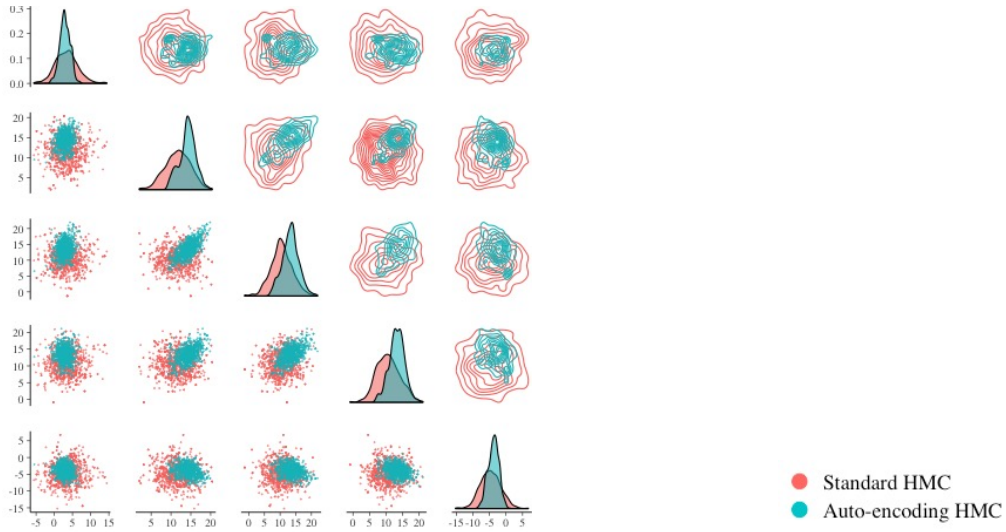


Figure 3.4: Posterior approximation: Standard HMC v.s. Auto-encoding HMC

3.4.3 Bayesian prediction for binary text classification

This experiment involves binary classification on benchmark dataset Quora question pairs ([35]). Our training data and test data includes 2,000 pairs of potential duplicate questions on Quora. The task is to identify the actual duplicate pairs. That being said, given two sentences, we have to determine if they have the same meaning. An example of a sentence pair could be "How do I read and find my YouTube comments?" and "How can I see all my

Youtube comments?". The expected output should be 1, which means duplicate. Another example is "What's causing someone to be jealous?", "What can I do to avoid being jealous of someone?". The expected output should be 0, which means they are not duplicate.

This is a text classification task and is high dimensional in nature. After preprocessing, such as word embedding and sentence embedding, each pair of sentences is represented as a 400-dimensional vector. We performed Bayesian logistic regression for classification. The predictions are based on posterior predictive distributions $p(\tilde{y}|y) = E_{p(\beta|y)}(p(\tilde{y}|\beta))$.

We run both standard HMC and Auto-encoding HMC with latent dimension 50 to obtain posterior samples of β. All the other settings are similar to the first experiment. We compared the prediction accuracy as the computational time evolves, as shown in Figure 3.5. The results of standard HMC eventually converges to a slightly higher accuracy compared to Auto-encoding HMC at around 30 seconds. However, in the first 10 seconds, Auto-encoding HMC achieves a higher prediction accuracy faster as it quickly converges.

3.5 Discussion

In this paper, we explored the possibility of simulating Hamiltonian dynamics in the latent space of the parameter space, but still making efficient exploration and distant proposals in the original space. The projection between the latent space and the original space is via an auto-encoder trained on pre-sampled data. The resulting algorithm Auto-encoding HMC achieves a good balance between computational resources and posterior approximation.

There are a set of future directions that worth pursuing.

First of all, our method lost some nice properties of standard HMC, such as volume preserving, and we need to evaluate an additional correction term. If a volume preserving embedding

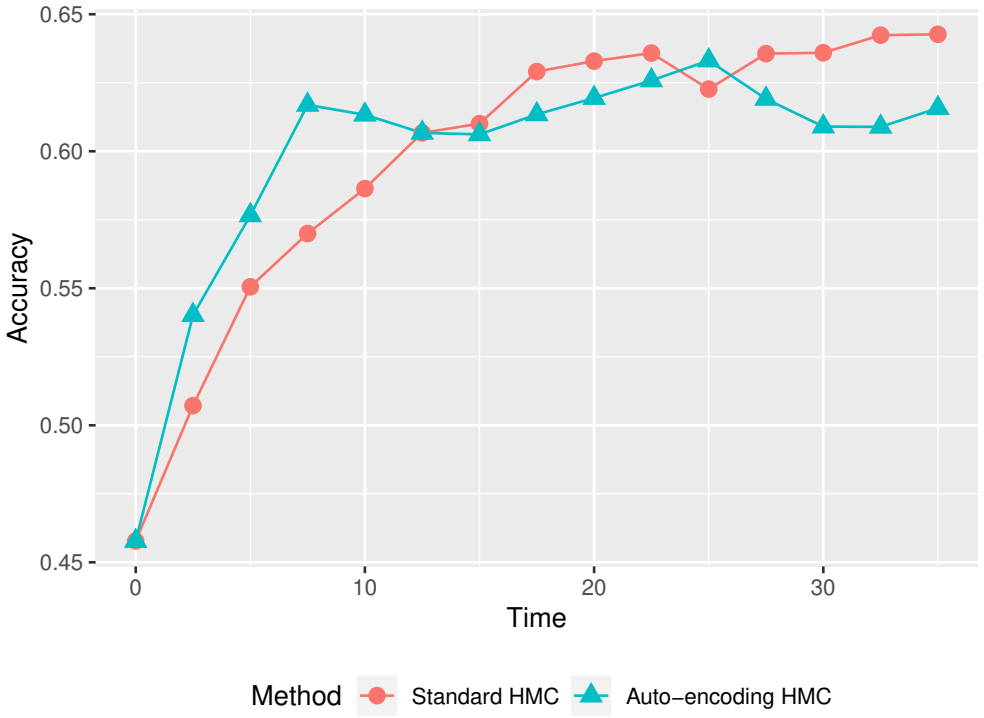


Figure 3.5: (Quora duplicate questions classification) Bayesian Prediction Accuracy against Computational Time

can be discovered, it will further save computational time. We also intend to find a better way to approximate the correction term.

The computational saving mainly depends on the dimension of the latent space, but less dimension will incur more information loss. More experiments need to be done to perform optimal tuning of the latent dimension. In addition, the design of auto-encoder architecture and selection of activation functions also need to be carefully tuned for better trade-off.

The experiments are still preliminary. We also need to move beyond logistic regression models.

We are also interested in comparing our method with a two-stage method in terms of prediction results. A two-stage method means we first perform standard HMC directly in the latent space and project the posterior samples to the original space for making Bayesian inference.

Also notice that our work can be extended to other MCMC algorithms using a similar

framework.

3.6 Supplementary

3.6.1 Calculating the gradient of $U_h(q_h)$ for Bayesian logistic regression

Consider a logistic regression model $y_i|\mathbf{X}_i, q_v \sim \text{Bern}(p_i), \quad p_i = \frac{1}{1+\exp(-\mathbf{X}_i q_v)}$, where $\mathbf{X}_i = (X_{i1}, X_{i2}, \cdots), \quad q_v = (q_{v1}, q_{v2}, \cdots)^T$. We set the prior to be zero mean Gaussian with unit variance.

The total Likelihood is $p(y|\mathbf{X}, q_v) = \prod_{i=1}^{N} \left(\frac{1}{1+\exp(-\mathbf{X}_i q_v)}\right)^{y_i} \left(\frac{1}{1+\exp(\mathbf{X}_i q_v)}\right)^{1-y_i}$

Given that the decoder has one hidden layer and *tanh* is used as the activation function, we have:

$$q_v = W_2 \tanh(W_1 q_h + b_1)$$

. Consider

$$\begin{aligned}
U_v(q_v) &= -\log(p(q_v)) - \log p(y|\mathbf{X}, q_v) \\
&= \frac{1}{2} q_v^T q_v - \sum_{i=1}^{N} \left(y_i \log \frac{1}{1+\exp(-\mathbf{X}_i q_v)} + (1-y_i) \log \frac{1}{1+\exp(\mathbf{X}_i q_v)} \right) \\
&= \frac{1}{2} q_v^T q_v - \sum_{i=1}^{N} y_i (\mathbf{X}_i q_v) + \sum_{i=1}^{N} \log(1+\exp(\mathbf{X}_i q_v)) \\
\frac{\partial U_v(q_v)}{\partial q_v} &= q_v - \mathbf{X}_{D\times N}^T \left(y - \frac{1}{1+\exp(-\mathbf{X} q_v)} \right)_{N\times 1} \\
&= q_v - \sum_{i=1}^{N} y_i \mathbf{X}_i^T + \sum_{i=1}^{N} \frac{1}{1+\exp(-\mathbf{X}_i q_v)} \mathbf{X}_i^T
\end{aligned}$$

Thus,

$$
\begin{aligned}
\frac{\partial U_h(q_h)}{\partial q_h} &= \frac{\partial U_v(q_v)}{\partial q_h} \\
&= \frac{\partial U_v(q_v)}{\partial q_v} \cdot \frac{\partial q_v}{\partial q_h} \\
&= \left\{ W_2 \tanh(W_1 q_h + b_1) - \sum_{i=1}^{N} y_i \mathbf{X}_i^T + \sum_{i=1}^{N} \frac{1}{1+\exp(-\mathbf{X}_i W_2 \tanh(W_1 q_h + b_1))} \mathbf{X}_i^T \right\}^T \\
&\quad \cdot W_2 \cdot diag(1 - \tanh^2(W_1 q_h + b_1)) \cdot W_1 \\
&= \left\{ \tanh(W_1 q_h + b_1)^T W_2^T W_2 - \sum_{i=1}^{N} y_i \mathbf{X}_i W_2 + \sum_{i=1}^{N} \frac{1}{1+\exp(-\mathbf{X}_i W_2 \tanh(W_1 q_h + b_1))} \mathbf{X}_i W_2 \right\} \\
&\quad \cdot diag(1 - \tanh^2(W_1 q_h + b_1)) \cdot W_1
\end{aligned}
\tag{3.6}
$$

where $W_2^T W_2$ and $X_i W_2$ can be pre-calculated. Notice that $\dim(X_i W_2) << \dim(X_i)$.

3.6.2 Time reversibility

Given that ψ is the inverse map of ϕ, and define function $h(q,p) = (q,-p)$ we have

$$
\begin{aligned}
\Psi(Q(\Phi(q_v, p_v))) &= \Psi(Q(\phi(q_v), \phi(p_v))) \\
&= \Psi(Q(q_h, p_h)) \\
&= \Psi(h \circ T_s \circ h(q_h, p_h)) \\
&= \Psi(h \circ T_s(q_h, -p_h)) \\
&= \Psi(h(q_h^*, -p_h^*)) \\
&= \Psi(q_h^*, p_h^*) \\
&= (\psi(q_h^*), \psi(p_h^*)) \\
&= (q_v^*, p_v^*)
\end{aligned}
$$

and

$$\begin{aligned}
\Psi(Q(\Phi(q_v^*, p_v^*))) &= \Psi(Q(\Psi^{-1}(q_v^*, p_v^*))) \\
&= \Psi(Q(\psi^{-1}(q_v^*), \psi^{-1}(p_v^*))) \\
&= \Psi(Q(q_h^*, p_h^*)) \\
&= \Psi(h \circ T_s \circ h(q_h^*, p_h^*)) \\
&= \Psi(h \circ T_s(q_h^*, -p_h^*)) \\
&= \Psi(h(q_h, -p_h)) \\
&= \Psi(q_h, p_h) \\
&= \Phi^{-1}(q_h, p_h) \\
&= (\phi^{-1}(q_h), \phi^{-1}(p_h)) \\
&= (q_v, p_v)
\end{aligned}$$

Thus $Q' = \Psi \circ Q \circ \Phi$ is symmetric.

3.6.3 Approximating volume correction term

Following [53], let's consider a one dimensional example. For mapping

$$T_\delta(q, p) = \begin{bmatrix} q \\ p \end{bmatrix} + \delta \begin{bmatrix} dq/dt \\ dp/dt \end{bmatrix} + O(\delta^2)$$

The Jacobian matrix:

$$B_\delta = \begin{bmatrix} 1 + \delta \dfrac{\partial^2 H}{\partial q \partial p} & \delta \dfrac{\partial^2 H}{\partial p^2} \\ -\delta \dfrac{\partial^2 H}{\partial q^2} & 1 - \delta \dfrac{\partial^2 H}{\partial p \partial q} \end{bmatrix} + O(\delta^2)$$

Consider a 3×2 matrix A and 2×3 matrix C:

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ a_{31} & a_{32} \end{bmatrix} \left\{ \begin{bmatrix} 1 + \delta \dfrac{\partial^2 H}{\partial q \partial p} & \delta \dfrac{\partial^2 H}{\partial p^2} \\ -\delta \dfrac{\partial^2 H}{\partial q^2} & 1 - \delta \dfrac{\partial^2 H}{\partial p \partial q} \end{bmatrix} + O(\delta^2) \right\} \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \end{bmatrix}$$

It gives a 3×3 matrix with element (i, j) to be

$$a_{i1}c_{1j} + a_{i2}c_{2j} + \delta(a_{i1}c_{1j}\frac{\partial^2 H}{\partial q \partial p} - a_{i2}c_{1j}\frac{\partial^2 H}{\partial p^2} + a_{i1}c_{2j}\frac{\partial^2 H}{\partial p^2} - a_{i2}c_{2j}\frac{\partial^2 H}{\partial p \partial q}) + O(\delta^2)$$

We could show that

$$\det(AB_\delta C) = \det(AC) + O(\delta^2)$$

The result can be generalized to higher dimensions.

Now let's denote $z = (q, p)$, $A_i = \dfrac{\partial z_v^i}{\partial z_h^i}$, $B_\delta = \dfrac{\partial z_h^i}{\partial z_h^{i-1}}$, $C_i = \dfrac{\partial z_h^i}{\partial z_v^i}$. We have:

$$\begin{aligned}
\det(\frac{\partial z_v^L}{\partial z_v^0}) &= \det(\frac{\partial z_v^L}{\partial z_h^L}\frac{\partial z_h^L}{\partial z_h^{L-1}}\frac{\partial z_h^{L-1}}{\partial z_v^{L-1}} \cdots \frac{\partial z_v^1}{\partial z_h^1}\frac{\partial z_h^1}{\partial z_h^0}\frac{\partial z_h^0}{\partial z_v^0}) \\
&= \det(A_L B_\delta C_{L-1} A_{L-1} \cdots A_1 B_\delta C_0) \\
&= \det(A_L B_\delta C_{L-1}) \det(A_{L-1} B_\delta C_{L-2}) \cdots \det(A_1 B_\delta C_0) \\
&= (\det(A_L C_{L-1}) + O(\delta^2))(\det(A_{L-1} C_{L-2}) + O(\delta^2)) \cdots (\det(A_1 C_0) + O(\delta^2)) \\
&= \frac{Vol(z_v^L)}{Vol(z_h^L)}\frac{Vol(z_h^0)}{Vol(z_v^0)} + O(\delta) \\
&\to \frac{Vol(z_v^L)}{Vol(z_h^L)}\frac{Vol(z_h^0)}{Vol(z_v^0)} \text{ as } \delta \to 0
\end{aligned}$$

For matrices A_L and C_0, the number of vectors are less than the dimension of the ambient space. We could use the square root of the gramian function of the matrix to calculate k-volume in n-space where $k < n$. In particular, for k linearly independent vectors $v_1, \cdots, v_k$,

the gramian function is $G(v1, ..., vk) = det(M^T M)$ where $M = (v_1, \cdots, v_k)$. The volume of the parallelepiped with the vectors is calculated by:

$$Vol(v1, ..., vk) = \sqrt{det(M^T M)}$$

Chapter 4

Determinantal Point Processes as Balancing Priors for Variational Auto-encoder

Variational auto-encoder (VAE) is widely used in latent representation learning and synthetic data generation. In the presence of class imbalance, the latent space will be redundant and dominated by the major class. In a conventional VAE, a standard normal prior is used for latent variable. In this paper, we propose to instead use a diversity encouraging prior — Determinantal Point Process prior, to 'up-weight' the minor class. Our method achieves higher minor class accuracy compared to the standard VAE in a multi-class imbalance case. It also generates more balanced synthetic data in a hand-written digits generation task.

4.1 Introduction

Class imbalance is a commonly encountered problem in real world data analysis, where some classes might be infrequent or rare so they are not represented in finite samples. As a result, the analysis might lead to biased estimates and inaccurate predictions. In supervised learning, the classification model might be dominated by the most frequent classes ignoring infrequent, but usually important classes. Imbalance can also affect data analysis in an unsupervised manner. For example, for latent variable models, the low dimensional representation of the data will be mainly dominated by the information collected on more frequently observed classes.

In this paper, we propose a method for alleviating class imbalance problem in latent variable models. A diversity encouraging point process — continuous k-DPP (determinantal point process with fixed cardinality k) is used as a prior for latent variables, which plays the role of regularizing the latent variables to be less redundant. In particular, we applied this prior to Variational Auto-encoder (VAE), which is one of the most widely used deep latent model.

We show that for imbalanced data, our method can improve the performance of both the representation learning model and the generative model. We evaluate our method on three problems. First, we show the advantages of our method over the conventional VAE based on an imbalanced MNIST classification task. We then use our method for neural decoding based on data from a Neurophysiological experiment, with the setting to be a multi-class, small sample size and high dimensional case. Finally, analysis for a image generation task using our methods is provided, which shows that the quality of the generative model is also enhanced by diversifying the latent variables.

The paper is organized as follows: in Section 4.2, we provide reviews for Variational Auto-encoder and Determinantal Point Process and its variants. We also review some existing methods for dealing with class imbalance. In Section 4.3, we describe our method in details.

Finally, in section 4.4, we provide results and analysis for three empirical experiments.

4.2 Background

4.2.1 Related Work

As summarized in [31], there are two major branches of methods for dealing with imbalanced learning problem: sampling methods and cost-sensitive methods. The methods are data-level and model-level respectively. The goal of imbalanced learning is usually to improve the performance of the underrepresented classes, even at the cost of slightly worse performance for the dominating classes.

Random oversampling and undersampling are the most popular sampling method since they are easy and straightforward to implement. However, undersampling has the problem of losing important information regarding the major class, while oversampling might will result in overfitting ([33], [43]).

Cost-sensitive methods usually associate a higher cost with misclassifying the minor class than misclassifying the major class ([21]). The methods are model-specific, and algorithms such as cost-sensitive boosting methods, decision trees and neural networks are developed ([31]).

Our method takes a probabilistic perspective, and thus is different from the above methods. Instead of 'up-weighting' the cost of misclassifying minor classes, our method intrinsically 'up-weight' the prior of the data's latent representation.

4.2.2 Variational auto-encoder

Variational auto-encoder (VAE) ([36]) is an unsupervised model for learning low-dimensional latent representation of a given dataset. It can also be used for learning deep generative models to generate realistic data such as images and texts. Compared to a standard auto-encoder, variational auto-encoder imposes a prior on the latent variable instead of treating it as a deterministic term. The prior can also be regarded as a regularizor on the latent variable.

More formally, we are interested in a set of data X and its latent representation Z. In the framework of graphical models, latent data z is generated from a pior $p_\theta(z)$, and data x is generated from model $p_\theta(x|z)$. VAE aims at efficient marginal inference for x as well as approximation for posterior distribution of z with a simplified model $q_\phi(z|x)$ ([36]). Unlike mean-field variational Bayes, VAE simultaneously maximizing marginal likelihood of x (evidence) and minimizing KL-divergence between $q_\phi(z|x)$ and $p_\theta(z|x)$. This is achieved by maximizing the evidence lower bound (ELBO) with respect to θ and ϕ:

$$\begin{aligned}\mathcal{L}(\theta, \phi; x) &= \log p_\theta(X) - KL(q_\phi(z|x) \| p_\theta(z|x)) \\ &= E_{q_\phi(z|x)}(\log p_\theta(x|z)) - KL(q_\phi(z|x) \| p_\theta(z))\end{aligned}$$

It's equivalent to minimizing $-E_{q_\phi(z|x)}(\log p_\theta(x|z))$, named as reconstruction loss, and minimizing KL-divergence between $q_\phi(z|x)$ and $p_\theta(z)$ at the same time.

The basic steps of a VAE involves passing data x into a encoder model to learn $q_\phi(z|x)$. Then latent variable z is sampled from $q_\phi(z|x)$ and fed into a decoder model and learn $p_\theta(x|z)$, which can be used to reconstruct new x. VAE involves learning both the inference model $q_\phi(z|x)$ and the generative model $p_\theta(x|z)$. The inference model $q_\phi(z|x)$, often referred to as an encoder, is a posterior distribution of the latent variable given the data. Meanwhile, the generative model $p_\theta(x|z)$, also referred to as a decoder, maps latent variables to a generative distribution for x. Both decoder and encoder models can be approximated by neural networks,

which is capable of learning most of the complex and nonlinear functions.

4.2.3 Determinantal Point Process (DPP) and k-DPP

Determinantal Point Process is a point process for modeling repulsion interactions between samples ([37]). It defines distribution over subsets of a fixed ground set, and assigns higher probability to more diverse subsets. In a discrete setting, suppose the ground set is $\mathcal{Y}$, $\mathcal{P}$ is defined to be a determinantal point process, if for every $A \subseteq \mathcal{Y}$,

$$\mathcal{P}(A \subseteq Y) \propto det(L_A)$$

where Y is a subset randomly drawn from $\mathcal{Y}$ according to $\mathcal{P}$. L is a kernel matrix: $\mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$, and L_A is its submatrix corresponding to all entries in A. For example, if $A = \{i, j\}$, where $i, j \in \mathcal{Y}$, then:

$$\mathcal{P}(A \subseteq Y) \propto det(L_A) = \begin{vmatrix} L_{ii} & L_{ij} \\ L_{ji} & L_{jj} \end{vmatrix}$$

Notice since the likelihood is proportional to the determinant of L_A, and thus the square of the volume spanned by the element vectors, the likelihood would be smaller for subsets with similar elements.

In a continuous setting, for ground $\Omega \subseteq \mathbb{R}^D$, similarly, we have a positive definite kernel function $L : \Omega \times \Omega \to \mathbb{R}$. For any $A \subseteq \Omega$, we have $P_L(A) \propto \det(L_A)$.

In some cases, it's necessary to fix the subset size for every drawn. A k-DPP is a determinantal

point process over subsets with cardinality k. For discrete setting, the likelihood is:

$$P_L(A) = \frac{det(L_A)}{\sum\limits_{|B|=k} det(L_B)} = \frac{det(L_A)}{e_k(\lambda_1, \cdots, \lambda_N)}$$

where $\lambda_1, \cdots, \lambda_N$ are eigenvalues of L and $e_k(\lambda_1, \cdots, \lambda_N)$ is the kth elementary symmetric polynomial ([1]). Similarly, for continuous setting we have:

$$P_L(A) = \frac{det(L_A)}{e_k(\lambda_{1:\infty})}$$

However, the term $e_k(\lambda_{1:\infty})$ is generally infeasible to evaluate.

4.3 Method

4.3.1 Balance latent variable with k-DPP prior

In a conventional VAE, a standard normal prior is used for latent variable z. We propose to instead use a continuous k-DPP prior for the continuous latent space, with the cardinality to be the sample size N. We then have the prior

$$p_\theta(z) = \mathcal{P}_L^N(z) = \frac{det(L_Z)}{e_N(\lambda_{1:\infty})}$$

where L_Z is a $N \times N$ kernel matrix.

Notice that the loss function of VAE is composed of reconstruction loss and KL-divergence (KLD) loss. For DPP-VAE, the reconstruction loss remains the same, while the KLD loss is

modified to be:

$$KL(q_\phi(z|x)\|p_\theta(z)) = \sum_{n=1}^{N}(-\log|\Sigma| - P) - E_{q_\phi(z|x)}(\log p_\theta(z))$$
$$\cong \sum_{n=1}^{N}(-\log|\Sigma| - P) - \ln\det(L_Z) + \ln(e_N(\lambda_{1:\infty})$$

To avoid large penalty from KLD loss, the approximating distribution $q_\phi(z|x)$ will be learned to generate more diverse, thus more balanced z across all classes. This is because when most samples are from the same class, in our case, the dominating class, $\det(L_Z)$ will be smaller since the samples from the same class will be more similar compared to when samples are from different classes.

4.3.2 Inference procedure

The other components of our model remain similar to the conventional VAE: the approximate posterior distribution for latent variable is parameterized as normal distribution: $q_\phi(z|x) = N(z|\mu(X), \Sigma(X))$. The generative model can be parametered as $p_\theta(x|z) = p_\theta(x|f(z))$. For example, if x is continuous, we can use $p_\theta(x|z) = N(x|f(z), I)$. If x is binary, we have $p_\theta(x|z) = Bern(x|f(z))$. Notice here $\mu(X)$, $\Sigma(X)$ and $f(z)$ are modeled by neural networks as nonlinear functions of X and ϕ, or Z and θ, where ϕ and θ are weights vectors in neural networks. The neural network structure of our method is displayed in Figure **??**.

Similar to the reconstruction loss, Monte Carlo estimates of expectations of $\log p_\theta(z)$ with respect to $q_\phi(z|x)$ is used here. In practice, we choose a positive definite kernel function

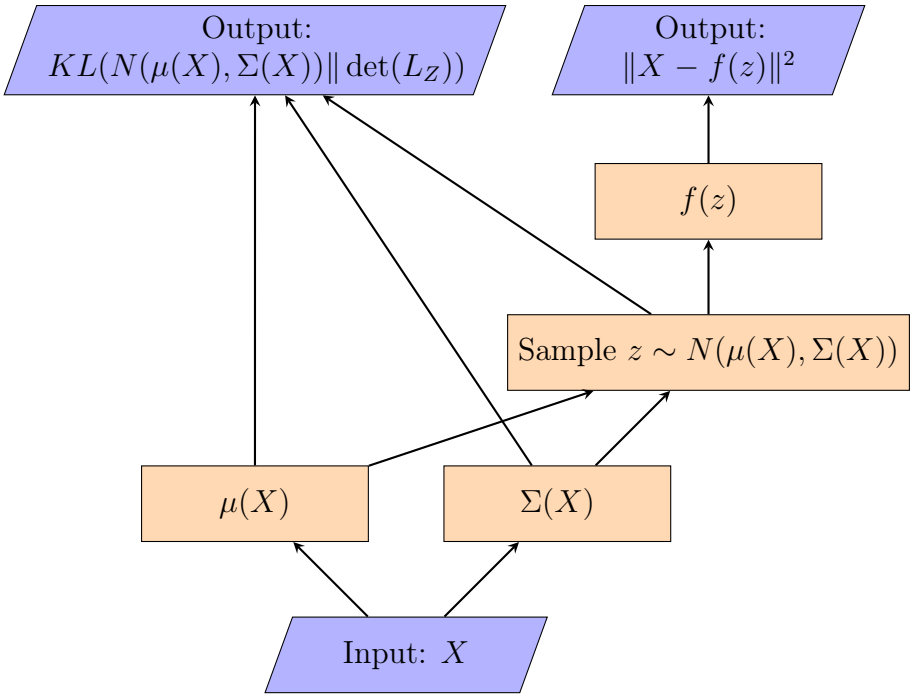


Figure 4.1: DPP-VAE Architecture (X is continuous)

$L(X)_{nm} = q(\mathbf{x}_n)k(\mathbf{x}_n, \mathbf{x}_m)q(\mathbf{x}_m)$ as suggested by ([1], [23]), where

$$q(x) = \sqrt{\alpha} \prod_{d=1}^{D} \frac{1}{\sqrt{\pi \rho_d}} \exp(-\frac{x_d^2}{2\rho_d})$$
$$k(x, y) = \prod_{d=1}^{D} \exp(-\frac{(x_d - y_d)^2}{2\sigma_d})$$

The eigenvalues can be obtained by:

$$\lambda_n = \alpha \prod_{d=1}^{D} (\frac{\beta_d^2 + 1}{2} + \frac{1}{2\gamma_d})^{-\frac{1}{2}} (\gamma_d(\beta_d^2 + 1) + 1)^{1-n_d}$$

Notice that the term $e_k(\lambda_{1:\infty})$ in KLD loss has no explicit form, but it's proved ([1]) that it

has lower bound and upper bound:

$$e_k(\lambda_{1:M}) <= e_k(\lambda_{1:\infty}) <= \sum_{j=0}^{k} \frac{(tr(L) - \sum_{n=1}^{M} \lambda_n)^j}{j!} e_{k-j}(\lambda_{1:M})$$

To maintain the KLD-loss to be positive, we can use the upper bound of the normalization term for approximation. $e_k(\lambda_{1:M})$ can be efficiently computed by algorithm developed in [37].

4.4 Experiments

4.4.1 Two-class MNIST data classification

We compared standard VAE and DPP-VAE on a binary classification task based on MNIST data. The training examples include 5000 MNIST '0', '1' handwritten digits data. For the balanced case, there are 2500 class 0 and 2500 class 1. We then varied the imbalance ratio from 1 to 1 to 1 to 1000, where digit '1' is the minor class. The test data is a balanced dataset with 500 class 0 and 500 class 1.

We select the latent dimension to be 20 and set $\alpha, \rho, \sigma = 1000, 1, 1$. All methods are trained for 10 epochs with batch size 100. Monte Carlo sample of size 1 is used since the batch size is large enough, as proposed in [36]. Two layer convolutional and deconvolutional neural network with ReLU activation is used for encoding and decoding the images separately. The latent samples are used for classification. We passed the learned features into a simple logistic regression model with optimal hyperparameters selected by cross-validation.

The performance of the balanced test data is displayed in Figure 4.2. The results are averaged over 20 independent runs. It shows that Variational Auto-encoder with continuous k-DPP

prior for imbalanced data always achieves a higher classification accuracy faster compared to standard VAE. The effect is more obvious as the imbalance ratio increases.

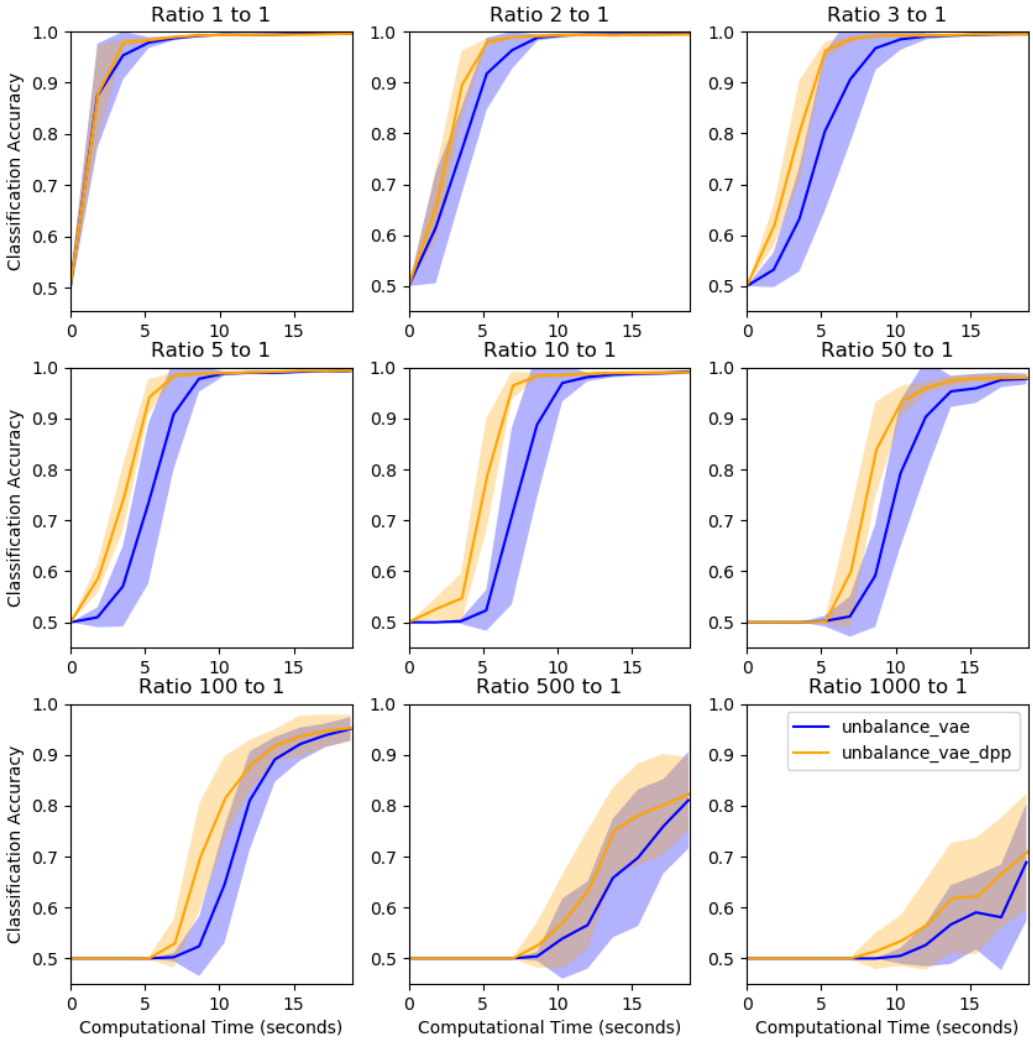


Figure 4.2: Comparing classification accuracy on balanced test data against computational time, as the imbalance ratio of training data evolves from 1 to 1 to 1000 to 1

4.4.2 Neural decoding: an application to multi-class imbalance

Most methods for imbalance learning focus on binary class imbalance. However, we also test our method on a multi-class problem, which is more challenging.

Many scientific studies face the challenge of collecting balanced datasets for high-quality data analysis. In a recent neuroscience experiment ([3]), rats are required to perform a sequence memory task and recognize five different odors in the given order A, B, C, D, E. We are interested in decoding their corresponding neural spike signals for odor classification. Neural decoding refers to the mapping from neural activities to stimulus. However, a sequence always started with odor A and will be terminated once the rat made a mistake. Thus there are more class A data than other classes, and decoding results are highly affected by this imbalance.

Data There are 58 trials for odor A, 41 trials for odor B, 37 trials for odor C, 32 trials for odor D and 26 trials for odor E. It's believed that a given odor is most related to the neural activities occurring during the 0.15s-0.4s time window after trial initiation. Thus, neural spike train data during that time window, which is mainly a binary sequence, for 54 isolated spikes is used for training.

Results The average performance based on a 6-fold cross-validation is displayed in Table 4.1 and ROC curves are provided in Figure 4.3, 4.4. The test data for each fold is selected to be a balanced set, with 4 test data for each class. The model is trained for 5000 epochs. Our proposed method enhanced the performance (e.g., f1 score) of the minor classes (C, D and E), even though at the cost of slightly lower performance for the major classes (A, B). And the overall performance is also improved.

Table 4.1: VAE and DPP-VAE comparison: Cross-validated performance

Table 4.2: VAE

	precision	recall	f1-score
A	0.706	0.875	0.776
B	0.636	0.625	0.621
C	0.233	0.417	0.294
D	0.215	0.167	0.172
E	0.139	0.083	0.103
ave	0.386	0.433	0.393

Table 4.3: DPP VAE

	precision	recall	f1-score
A	0.751	0.917	0.809
B	0.497	0.708	0.575
C	0.361	0.458	**0.377**
D	0.333	0.250	**0.278**
E	0.333	0.083	**0.133**
ave	0.455	0.483	**0.434**

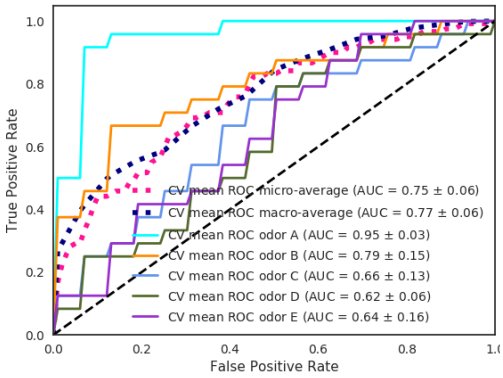


Figure 4.3: Standard VAE

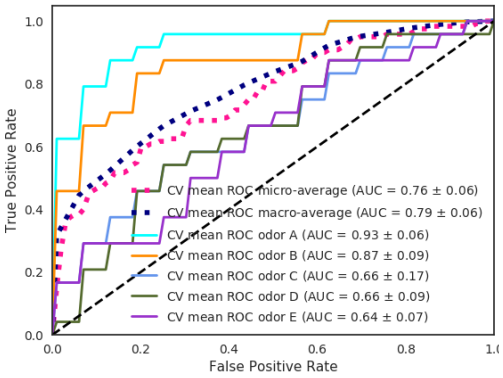


Figure 4.4: DPP VAE

4.4.3 Balancing data generation

In this experiment, variational auto-encoders are trained to draw MNIST digits. We showed the effectiveness of DPP-VAE for balancing the class ratios of the generated data.

The parameter settings are the same as the first experiment. Synthetic MNIST data (digit '0' and '1') are generated by standard VAE and DPP-VAE separately. Random latent vectors from standard normal distribution are fed into the trained decoder network to generate handwritten '0's and '1's. The same latent vectors are used for both methods. The total training data is 5000 and we tested on 3 different imblance ratios: 10 to 1, 100 to 1 and 1000 to 1.

Results are displayed in Table 4.4. For standard VAE, the generated minor class (digit

'1') percentage is around the same as that of the training data, which means the learned decoder is dominated by the major class. However, by implementing our proposed method, the percentage of the minor class is substantially increased, and the effect is more significant as the imbalance ratio increases: 1.9 times increase for ratio 10 to 1 (17.7% versus 9.1%), 3.7 times increase for ratio 100 to 1 (3.68% versus 0.99%), and 9.5 times increase for ratio 1000 to 1 (0.9469% versus 0.0999%).

Table 4.4: Generated minor class (digit '1') percentage

Class ratio	Training (%)	VAE (%)	DPP-VAE (%)
10:1	9.1%	7.2%	**17.7%**
100:1	0.99%	1.21%	**3.68%**
1000:1	0.0999%	0.0562%	**0.9469%**

We also visualized 900 synthetic data with training ratio 10 to 1 in Figure 4.5, 4.6, . The digits are ordered by classes. The class of the generated data is decided by a highly accurate convolutional neural network trainer.

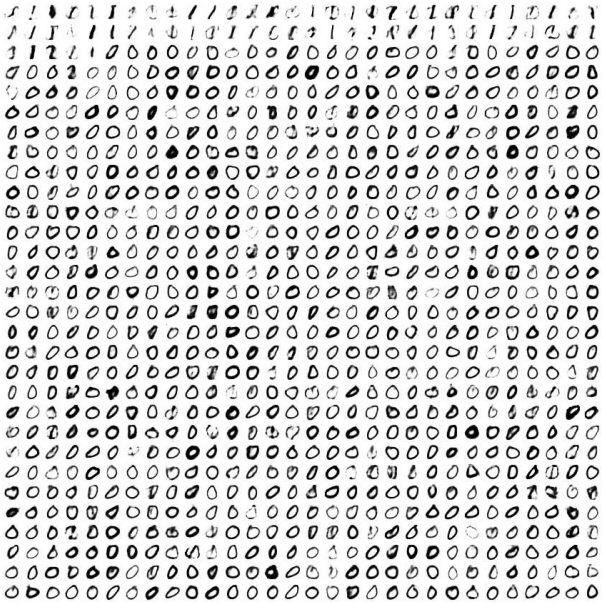


Figure 4.5: Standard VAE

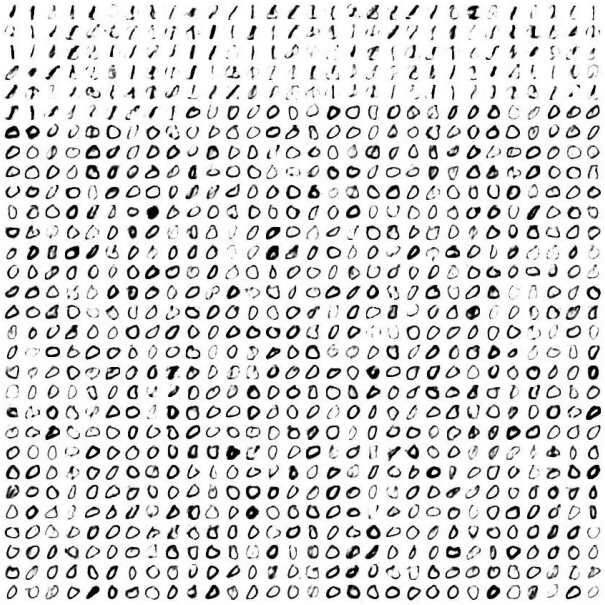


Figure 4.6: DPP VAE

4.5 Discussion

In this work, we propose to use Determinantal Point Process as a diversity encouraging prior for latent variable models to alleviate imbalance learning problem . We have a particular application, where we modified variational auto-encoder by using continuous k-DPP as latent prior, and developed the inference algorithm. Our proposed method improved the minor class performance compared to standard VAE in binary and multi-class classification tasks. DPP-VAE is also capable of generating more balanced synthetic data.

Our work has significant contributions to many machine learning applications, especially when samples of rare classes with great importance are hard to collect. For example, in self-driving perceptron tasks, it's expensive and time-consuming to obtain enough ground-truth training samples for rare and extreme weather conditions. One possible solution is to generate as many synthetic data of rare environments as possible given imbalanced training classes.

Chapter 5

Conclusions

5.1 Summary

Bayesian inference typically involves intractable integration with respect to the posterior distribution of model parameters or latent variables. We usually resort to either stochastic approximation, which is based on sampling methods, or deterministic approximation, which is based on functional optimization. In practice, sampling methods are commonly applied to smaller problems when high precision is required, while variational methods are usually used for large applications when only a quick approximation is needed.

We are faced with new computational challenges for Bayesian inference due to the emergence of data intensive problems. Thus, research on new variational methods or scalable sampling methods is crucial. To this end, we have used modern information geometry as a powerful tool for statistical analysis in order to provide a geometric view for these approximation methods.

We first developed a new variational framework based on differential geometry: geometric

approximation of posterior (GAP), where we proposed a valid distance measure — spherical Fisher distance and developed modified gradient descent algorithm to minimize the distance. Compared to other variational methods, under certain scenarios, GAP generates better approximating distributions. Compared to Hellinger distance, geometrically, using the spherical Fisher distance is more justifiable and the optimization is more smooth.

For certain large applications, however, we might need more precision, and thus we have developed auto-encoding HMC as a scalable sampling method. We explored the idea of simulating Hamiltonian dynamics in the latent space of the parameter space, but still making efficient exploration in the original space. Auto-encoding HMC achieves a good balance between computational cost and accuracy.

Aside from developing Bayesian computational methods, we also modified classification models for tackling imbalanced learning problems, motivated by a real application of neural decoding with imbalanced classes. We then developed DPP-VAE, where we applied determinantal point process as a diversity encouraging prior for variational auto-encoder. Our method improved the minority class performance for classification and synthetic data generation.

5.2 Future directions

While GAP method achieves better results for multi-modal problems than other variational methods, we limit the approximating distribution within the family of Gaussian distributions. A future work can address this issue by developing more flexible proposals, for example, using mixture of Gaussians or Dirichlet process mixture models.

Another research direction could be improving computational efficiency of our method. Currently, the computational cost is dominated by orthonormalizing the basis of the tangent space using Gram-Schmidt Matrix representation, which is computationally demanding.

Additionally, there is a high computational cost associated with approximating intractable integrals involved in calculating the inner products between functions. To evaluate these integrals, we need to sample from tractable posterior approximating distribution and use Monte Carlo estimates.

We also would like to work on identifying natural classes of convex and non-convex models, since gradient descent has the problem of getting trapped in local minimum. For non-convex problems, we can then develop randomized methods to escape from the local minimum.

We also need to further improve our Auto-encoding HMC method. For example, we could use empirical KL-divergence as a metric for measuring posterior approximation results. We also plan to experiment with different parameter, hidden and latent dimensions to find an optimal trade-off between computational efficiency and precision. It is also potentially an important work to quantify the errors introduced by our method, possibly using universal approximation theorem. Lastly, it is worth extending our method to other MCMC algorithms.

Regarding our DPP-VAE method, which involves a deep generative model, we could expand our work by applying it to more complicated tasks with higher dimensions, such as generating synthetic texts and real-world images. We also would like to investigate the computational complexity of DPP-VAE and how well it scales to high dimensional problems.

Bibliography

[1] R. H. Affandi, E. Fox, R. Adams, and B. Taskar. Learning the parameters of determinantal point process kernels. In *International Conference on Machine Learning*, pages 1224–1232, 2014.

[2] Y. Ahmadian, J. W. Pillow, and L. Paninski. Efficient Markov Chain Monte Carlo methods for decoding neural spike trains. *Neural Computation*, 23(1):46–96, 2011.

[3] T. A. Allen, D. M. Salz, S. McKenzie, and N. J. Fortin. Nonspatial sequence coding in ca1 neurons. *Journal of Neuroscience*, 36(5):1547–1563, 2016.

[4] S. Amari and H. Nagaoka. *Methods of Information Geometry*, volume 191 of *Translations of Mathematical monographs*. Oxford University Press, 2000.

[5] S. Amari and H. Nagaoka. *Methods of Information Geometry*. Translations of mathematical monographs. American Mathematical Society, 2007.

[6] C. Andrieu and E. Moulines. On the ergodicity properties of some adaptive mcmc algorithms. *Annals of Applied Probability*, 16(3):1462–1505, 2006.

[7] M. J. Beal. *Variational Algorithms for Approximate Bayesian Inference*. PhD thesis, University College London, London, UK, 2003.

[8] A. Beskos, N. Pillai, G. Roberts, J.-M. Sanz-Serna, A. Stuart, et al. Optimal tuning of the hybrid monte carlo algorithm. *Bernoulli*, 19(5A):1501–1534, 2013.

[9] A. Beskos, F. J. Pinski, J. M. Sanz-Serna, and A. M. Stuart. Hybrid Monte-Carlo on Hilbert spaces. *Stochastic Processes and their Applications*, 121:2201–2230, 2011.

[10] M. Betancourt. A conceptual introduction to hamiltonian monte carlo. *arXiv preprint arXiv:1701.02434*, 2017.

[11] C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006.

[12] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877, 2017.

[13] A. E. Brockwell. Parallel markov chain monte carlo simulation by Pre-Fetching. *Journal of Computational and Graphical Statistics*, pages 246–261, 2006.

[14] B. Calderhead and M. Sustik. Sparse approximate manifolds for differential geometric mcmc. In P. Bartlett, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 2888–2896. 2012.

[15] O. Cappé, R. Douc, A. Guillin, J.-M. Marin, and C. P. Robert. Adaptive importance sampling in general mixture classes. *Statistics and Computing*, 18(4):447–459, 2008.

[16] R. V. Craiu, J. R., and C. Y. Learn from thy neighbor: Parallel-chain and regional adaptive mcmc. *Journal of the American Statistical Association*, 104(488):1454–1466, 2009.

[17] B. C. Csáji. Approximation with artificial neural networks. *Faculty of Sciences, Etvs Lornd University, Hungary*, 24:48, 2001.

[18] A. P. Dawid et al. Further comments on some comments on a paper by bradley efron. *The Annals of Statistics*, 5(6):1249–1249, 1977.

[19] N. de Freitas, P. Højen-Sørensen, M. Jordan, and R. Stuart. Variational MCMC. In *Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence*, UAI '01, pages 120–127, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[20] C. Doersch. Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908*, 2016.

[21] C. Elkan. The foundations of cost-sensitive learning. In *International joint conference on artificial intelligence*, volume 17, pages 973–978. Lawrence Erlbaum Associates Ltd, 2001.

[22] P. L. Fackler. Notes on matrix calculus. *Privately Published*, 2005.

[23] G. E. Fasshauer and M. J. McCourt. Stable evaluation of gaussian radial basis function interpolants. *SIAM Journal on Scientific Computing*, 34(2):A737–A762, 2012.

[24] X. Gao, B. Shahbaba, N. Fortin, and H. Ombao. Evolutionary state-space model and its application to time-frequency analysis of local field potentials. *arXiv preprint arXiv:1610.07271*, 2016.

[25] X. Gao, W. Shen, and H. Ombao. Regularized matrix data clustering and its application to image analysis. *arXiv preprint arXiv:1808.01749*, 2018.

[26] X. Gao, W. Shen, C.-M. Ting, S. C. Cramer, R. Srinivasan, and H. Ombao. Modeling brain connectivity with graphical models on frequency domain. *arXiv preprint arXiv:1810.03279*, 2018.

[27] A. Gelfand, L. van der Maaten, Y. Chen, and M. Welling. On herding and the cycling perceptron theorem. In *Advances in Neural Information Processing Systems 23*, pages 694–702, 2010.

[28] C. J. Geyer. Practical Markov Chain Monte Carlo. *Statistical Science*, 7(4):473–483, 1992.

[29] W. R. Gilks, G. O. Roberts, and S. K. Sahu. Adaptive markov chain monte carlo through regeneration. *Journal of the American Statistical Association*, 93(443):1045–1054, 1998.

[30] M. Girolami and B. Calderhead. Riemann manifold Langevin and Hamiltonian Monte Carlo methods. *Journal of the Royal Statistical Society, Series B*, (with discussion) 73(2):123–214, 2011.

[31] H. He and E. A. Garcia. Learning from imbalanced data. *IEEE Transactions on knowledge and data engineering*, 21(9):1263–1284, 2009.

[32] M. Hoffman and A. Gelman. The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo. arxiv.org/abs/1111.4246, 2011.

[33] R. C. Holte, L. Acker, B. W. Porter, et al. Concept learning and the problem of small disjuncts. In *IJCAI*, volume 89, pages 813–818. Citeseer, 1989.

[34] T. S. Jaakkola and M. I. Jordan. Bayesian parameter estimation via variational methods. *Statistics and Computing*, 10(1):25–37, 2000.

[35] Kaggle. Quora question pairs. `https://www.kaggle.com/c/quora-question-pairs`, 2017.

[36] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[37] A. Kulesza, B. Taskar, et al. Determinantal point processes for machine learning. *Foundations and Trends® in Machine Learning*, 5(2–3):123–286, 2012.

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

[39] S. Lan, B. Zhou, and B. Shahbaba. Spherical Hamiltonian Monte Carlo for Constrained Target Distributions. In *31th International Conference on Machine Learning (to appear)*, 2014.

[40] J. Lee. *Introduction to Smooth Manifolds*. Graduate Texts in Mathematics. Springer, 2003.

[41] L. Li, A. Holbrook, B. Shahbaba, and P. Baldi. Neural network gradient hamiltonian monte carlo. *arXiv preprint arXiv:1711.05307*, 2017.

[42] D. J. C. MacKay. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2003.

[43] D. Mease, A. J. Wyner, and A. Buja. Boosted classification trees and class probability/quantile estimation. *Journal of Machine Learning Research*, 8(Mar):409–439, 2007.

[44] T. Minka et al. Divergence measures and message passing. Technical report, Technical report, Microsoft Research, 2005.

[45] T. P. Minka. Expectation propagation for approximate bayesian inference. In *Proceedings of the Seventeenth conference on Uncertainty in artificial intelligence*, pages 362–369. Morgan Kaufmann Publishers Inc., 2001.

[46] J. Møller, A. Pettitt, K. Berthelsen, and R. Reeves. An efficient Markov chain Monte Carlo method for distributions with intractable normalisation constants. *Biometrica*, 93, 2006. to appear.

[47] P. Mykland, L. Tierney, and B. Yu. Regeneration in Markov Chain Samplers. *Journal of the American Statistical Association*, 90(429):233–241, 1995.

[48] R. M. Neal. *Probabilistic Inference Using Markov Chain Monte Carlo Methods*. Technical Report CRG-TR-93-1, Department of Computer Science, University of Toronto, 1993.

[49] R. M. Neal. Sampling from multimodal distributions using tempered transitions. *Statistics and Computing*, 6(4):353, 1996.

[50] R. M. Neal. Slice sampling. *Annals of Statistics*, 31(3):705–767, 2003.

[51] R. M. Neal. The short-cut metropolis method. Technical Report 0506, Department of Statistics, University of Toronto, 2005.

[52] R. M. Neal. MCMC using Hamiltonian dynamics. In S. Brooks, A. Gelman, G. Jones, and X. L. Meng, editors, *Handbook of Markov Chain Monte Carlo*, pages 113–162. Chapman and Hall/CRC, 2011.

[53] R. M. Neal et al. Mcmc using hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*, 2(11):2, 2011.

[54] K. B. Petersen, M. S. Pedersen, et al. The matrix cookbook. *Technical University of Denmark*, 7(15):510, 2008.

[55] J. G. Propp and D. B. Wilson. Exact sampling with coupled Markov chains and applications to statistical mechanics. volume 9, pages 223–252, 1996.

[56] D. Randal and C. R. P. A vanilla rao-blackwellization of metropolis-hastings algorithms. *Annals of Statistics*, 39(1):261–277, 2011.

[57] R. Randal, G. Arnaud, M. Jean-Michel, and R. P. Christian. Minimum variance importance sampling via population monte carlo. *ESAIM: Probability and Statistics*, 11:427–447, 2007.

[58] G. O. Roberts and S. K. Sahu. Updating Schemes, Correlation Structure, Blocking and Parameterisation for the Gibbs Sampler. *Journal of the Royal Statistical Society, Series B*, 59:291–317, 1997.

[59] B. Shahbaba, S. Lan, W. Johnson, and R. Neal. Split Hamiltonian Monte Carlo. *Statistics and Computing*, 24(3):339–349, 2014.

[60] B. Shahbaba, S. Lan, W. O. Johnson, and R. M. Neal. Split hamiltonian monte carlo. *Statistics and Computing*, 24(3):339–349, 2014.

[61] G. R. Warnes. The normal kernel coupler: An adaptive Markov Chain Monte Carlo method for efficiently sampling from multi-modal distributions. Technical Report Technical Report No. 395, University of Washington, 2001.

[62] M. Welling. Herding dynamic weights to learn. In *Proc. of Intl. Conf. on Machine Learning*, 2009.

[63] M. Welling and Y. Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th International Conference on Machine Learning (ICML)*, pages 681–688, 2011.

[64] C. Zhang, B. Shahbaba, and H. Zhao. Hamiltonian monte carlo acceleration using surrogate functions with random bases. *Statistics and computing*, 27(6):1473–1490, 2017.

[65] C. Zhang, B. Shahbaba, and H. Zhao. Precomputing strategy for hamiltonian monte carlo method based on regularity in parameter space. *Computational Statistics*, 32(1):253–279, 2017.

[66] Y. Zhang and C. Sutton. Quasi-Newton Markov chain Monte Carlo. In *Advances In Neural Information Processing Systems*, 2011.

[67] H. Zhu and R. Rohwer. Information geometric measurements of generalisation. 1995.

Appendix A

Deep Learning for Rat's Sequence Memory Replay

A.1 Experiment Description

The experiment consists of five rats across three sessions. In the following analysis, we only focus on the data from one rat 'SuperChris' collected during session 1.

Sequence Memory Task The core task for rats is a sequential memory task. A sequence of different odors are presented to rats. Each sequence contains up to five trials, each of which lasts no more than 2 seconds. Each trial involves one of five odors, which are named 'A', 'B', 'C', 'D', 'E' respectively. An odor is "In Sequence" (InSeq) if it occurs in the right position during the sequence. For example, if odor C occurs during the third trial of a sequence it is "In Sequence". If odor B occurs during fourth trial it is then "Out of Sequence" (OutSeq).

Here are some examples:

- ABCDE (all InSeq)
- AACDE (second trial OutSeq, the rest InSeq)
- ABDDE (third trial OutSeq)

Rat's Performance Each trial starts with the rat poking its nose into a port, and an odor is immediately released into the port. The rat has to decide if it's InSeq or OutSeq. If it's InSeq, the rat has to hold its nose inside the port for more than 1.2 second. If it's OutSeq, the rat has to withdraw its nose within 1.2 second. A sequence will be terminated once the rat made a mistake, so some sequences contain fewer than five trials. For session 1 of SuperChris, there are 60 sequences with 248 trials in total, and the session last 50 minutes.

Neural Activities Data In each session, the neural activities of each rat are continuously recorded, including neural spike train data (55 isolated spikes) and local field potentials (12 tetrodes). Theta and Slow gamma signals are different types of local field potentials (LFP). The neural spike train data corresponds to the time of spikes for each neuron.

A.2 Sequence Replay Modeling

Many work has been done to analyze the neural spike train and LFP data from the experiments ([24, 25, 26, 26]). However, the fundamental question here is sequence memory replay. In particular, whether the neural activities corresponding to the five odors are replayed in the sequence of ABCDE or not during the session. We define neural states and associate them with odors. Here, we name the following processes:

- Neural Encoding: mapping from stimulus to neural activities. We study how neurons respond to stimulus, and construct models to predict neural activities.

- Neural Decoding: reverse mapping from neural activities to stimulus. We build models to reconstruct stimulus from neural activities.

Model $f(y|X)$ where X represents neural activity and y are classes of odors. In our experiment, we believe that a given odor is most closely related to the neural activities occurring during the 0.15 second to 0.4 second time window after nose-poke. In the framework of supervised classification, we use the 0.15s to 0.4s neural activities as training data and odors as different classes, and predict/reconstruct odors using neural activities during other times of the trials.

Examine sequence replay during trials For now, we are expecting to identify some sort of replay during the -2s to 2s time window relative to the nose-poke, and we are using the neural activities during this time period as test data. With a 0.02 second stride ,the test windows are: $[-2, -1.75], [-1.98, -1.73], ..., [0, 0.25], [0.02, 0.27], ...[1.75, 2]$

Moe data details There are 55 neurons and 248 trials in total. If we only consider in sequence and correct trials, there are 194 in total. We are also faced with imbalanced classes. Out of 194 in sequence and correct trials, we have 58 trials for A, 41 for B, 37 for C, 32 for D and 26 for E. This is because rats tend to memorize the first few odors, and make mistakes later during a sequence.

A.3 Prediction Results Using Logistic Regression, CNN, VAE and DPP-VAE

Our baseline model is logistic regression model, where we used the sum of spike counts during the training window as the training data. The prediction results are displayed in Figure A.1.

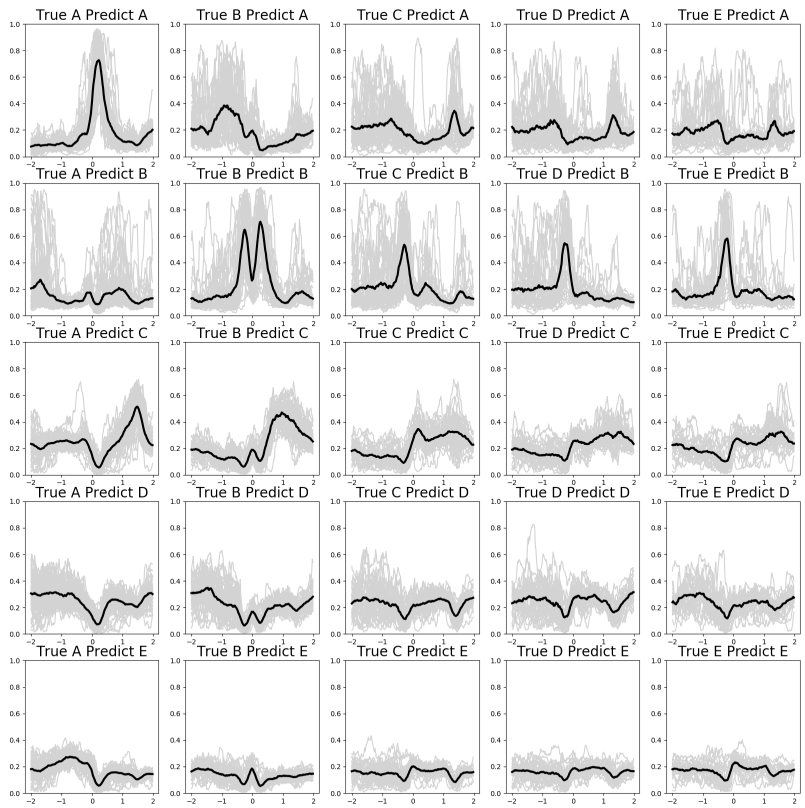


Figure A.1: Sequence replay predictions using logistic regression

We also consider using Convolutional Neural Network (CNN) to approach the problem. One challenge we have is that for each trial, even though we focus on the 0.15s to 0.4s time window, the neural state corresponding to a specific odor can occur at any time within the window and also occur at different time scales (e.g., some could be happening within a shorter time frame). CNN can work better than logistic regression since it's invariant to time shift of the signals and can potentially help find the 'local signals'.

Prediction results by implementing an LeNet (a type of CNN) on the dataset are displayed in Figure A.2. We only used neural spike train as training data. The 0.15s-0.4s time window is binned every 0.01s, and the number of spikes is accumulated for each bin, so the input is a sequence of number of spikes. Then a CNN filter will scan through this 1×250 sequence. The predictions are averaged over all trials for the -2s to 2s time window. Notice that in our results, the prediction is highly accurate for time window 0.15s-0.4s, which is the training data, which means we should be careful about potential overfitting issues.

We also display and compare corresponding results using standard VAE and DPP-VAE in Figure A.3 and A.4. The methods are described in Chapter 4.

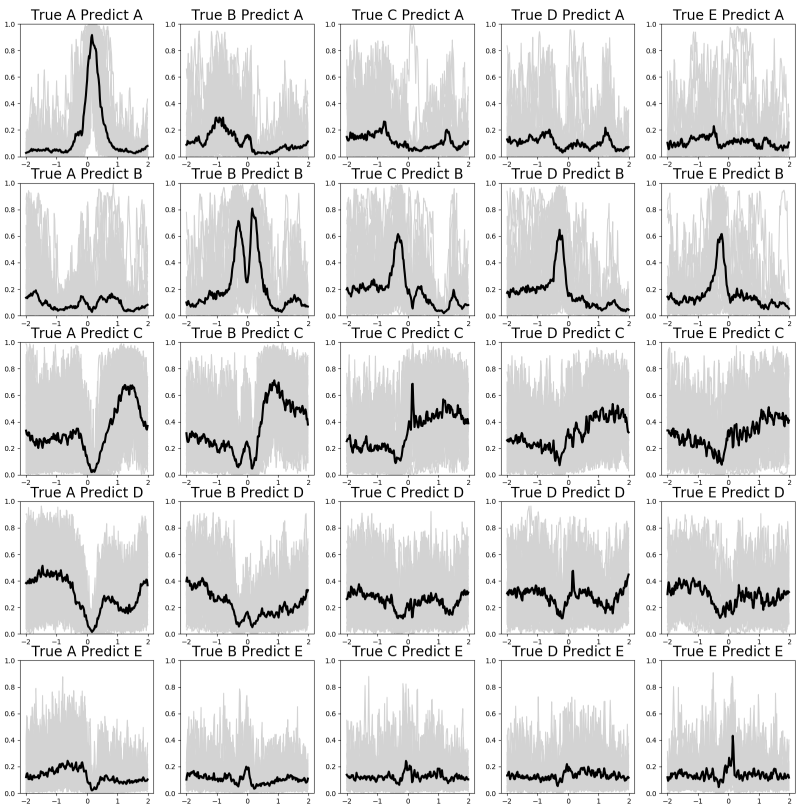


Figure A.2: Sequence replay predictions using convolutional neural network

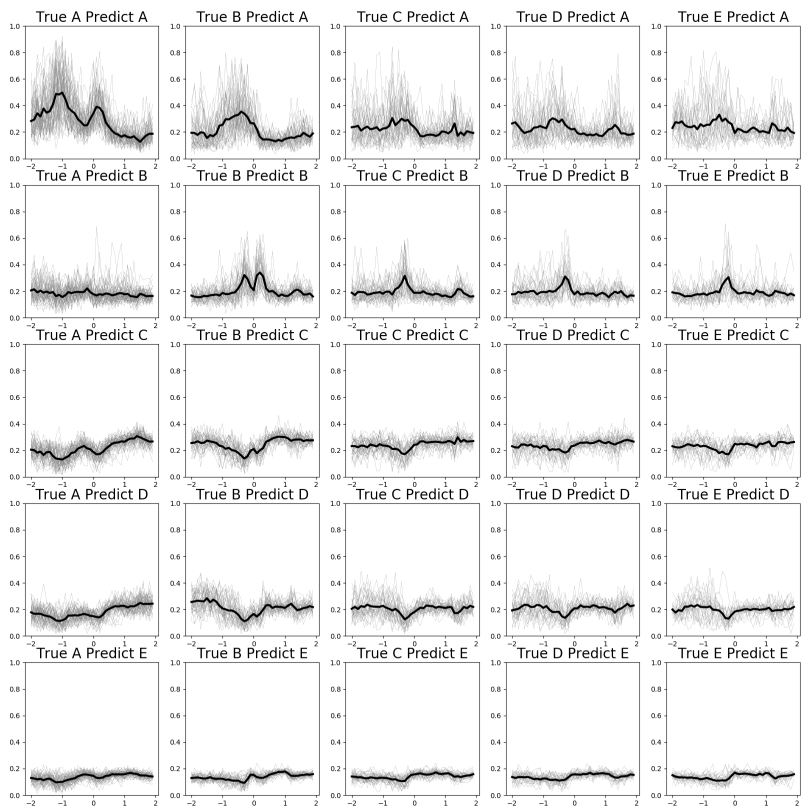


Figure A.3: Sequence replay predictions using variational auto-encoder

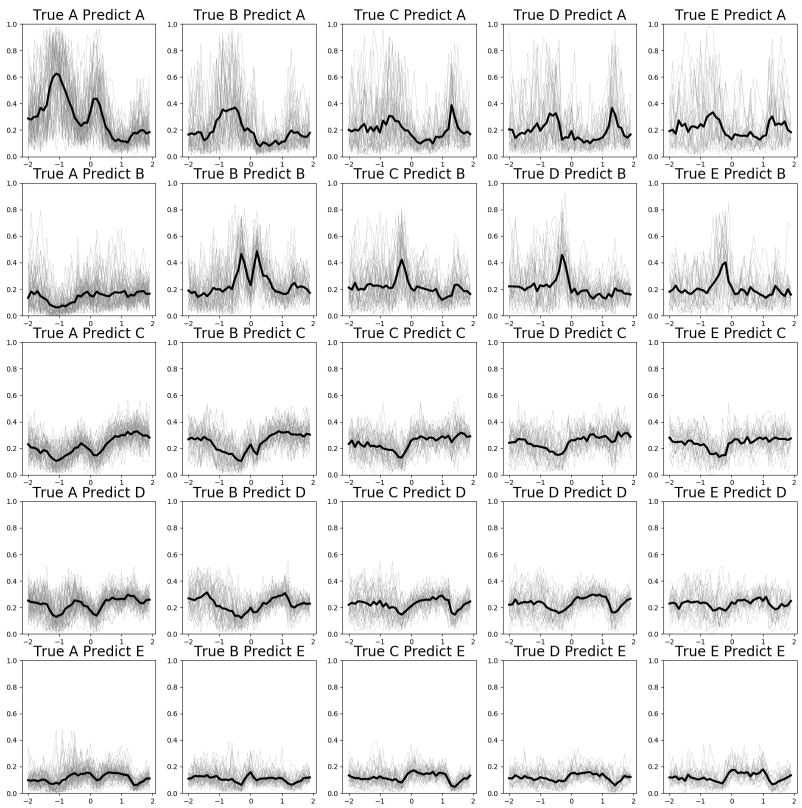


Figure A.4: Sequence replay predictions using DPP-VAE