Scalable Methodologies for Optimizing Over Probability Distributions

by

Lingxiao Li

B.S., Stanford University, 2018
M.S., Stanford University, 2019

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2024

Authored by: Lingxiao Li
Department of Electrical Engineering and Computer Science
May 17, 2024

Certified by: Justin Solomon
Associate Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by: Leslie A. Kolodziejski
Professor of Electrical Engineering and Computer Science
Chair, Department Committee on Graduate Students

Scalable Methodologies for Optimizing Over Probability Distributions

by

Lingxiao Li

Submitted to the Department of Electrical Engineering and Computer Science
on May 17, 2024 in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

ABSTRACT

Modern machine learning applications, such as generative modeling and probabilistic inference, demand a new generation of methodologies for optimizing over the space of probability distributions, where the optimization variable represents a weighted population of potentially infinitely many points. Despite the ubiquity of these distributional optimization problems, there has been a shortage of scalable methods grounded in mathematical principles. To bridge this gap, this thesis introduces two complementary lines of works for scalable distributional optimization.

The first part of this thesis focuses on optimizing over discrete distributions to generate high-quality samples for probabilistic inference. We present two works that tackle sampling by optimizing pairwise interaction energies defined on a collection of particles. The first work focuses on designing a new family of mollified interaction energies over moving particles, offering a unified framework for constrained and unconstrained sampling. The second work focuses on the scalable optimization of a family of popular interaction energies—maximum mean discrepancy of mean-zero kernels—to generate high-quality coresets from millions of biased samples, obtaining better-than-i.i.d. unbiased coresets.

The second part transitions to optimizing over continuous distributions through neural network parameterization, enabling the generation of endless streams of samples once optimized. We exploit convexity principles to identify suitable mathematical formulations and scalable optimization algorithms in three contexts: 1) averaging distributions in a geometric meaningful manner using a regularized Wasserstein barycenter dual formulation; 2) identifying local minima of non-convex optimization as a generative model by learning proximal operators with global convergence guarantees; and 3) solving mass-conserving differential equations of probability flows without temporal or spatial discretization by leveraging the self-consistency of the dynamical system.

Thesis supervisor: Justin Solomon
Title: Associate Professor of Electrical Engineering and Computer Science

Acknowledgments

First and foremost, I extend my deepest gratitude to my PhD advisor, Justin Solomon, for his unwavering support over the years. I first met Justin almost a decade ago when I took his numerical analysis class at Stanford. Despite rarely attending in person, I was very impressed by his enthusiasm across the computer screen. At the end of the semester, I had a chance to chat with him about career advice. While the specifics of that conversation have faded, his sincerity and passion really stuck with me. For some reason, every year afterwards, he agreed to meet with me, offering unreserved advice, whether at SIGGRAPH or at his MIT office, long before I became his responsibility. Looking back, I am profoundly grateful to have become his student and together we have explored a diverse set of research topics, many of which culminated in this thesis. I'm always impressed by his kindness, optimism, creativity, and relentless effort on outreach.

I am tremendously thankful to my other committee members for their mentorship and contributions. Ashia Wilson, who joined the committee merely ten minutes after my request, brings wonderful positivity and expertise in optimization and statistics. Lester Mackey, my mentor during a summer internship at Microsoft Research, has been nothing short of magical. His knack for solving problems and attention to detail have significantly shaped my research skills. I feel lucky to have worked with him closely on nearly all aspects of the project, from pinning down constants in theorems to painstakingly crafting the prose.

I owe a great deal to a diverse group of collaborators who have been integral to my projects: Mazdak Abulnaga, Noam Aigerman, Evgeny Burnaev, Anastasia Dubrovina, Raaz Dwivedi, Vage Egiazarian, Alexander Filippov, Aude Genevay, Kristjan Greenewald, Leonidas Guibas, Samuel Hurault, Vladimir G. Kim, Anna Korba, Alexander Korotin, Jiajin Li, Qiang Liu, Lester Mackey, Petr Mokrov, Dmitriy Smirnov, Justin Solomon, Minhyuk Sung, Li Yi, Mikhail Yurochkin, and Paul Zhang. Many thanks to all of you for making each of my projects a reality.

In this journey, my family in China has been my cornerstone. Ever since childhood, they have provided unconditional support and freedom to explore my dreams and passions. For example, they were supportive when I dropped out of the first college and joined a

video game startup, and they showed immediate acceptance when I came out to them early in college. I am very lucky to have them as my parents.

I would also like to thank my partner, Austin McCredie, and his family, for making the last few years much more colorful. I feel grateful to have a second home here in the US and I look forward to our future together.

I am extremely grateful to have supportive and fun-loving labmates with whom I have crossed paths: Mazdak Abulnaga, Edward Chien, Erica Chiu, Sebastian Claici, Daryl DeFord, Ana Dodik, Victor Dorobantu, Rickard Brüel Gabrielsson, Aude Genevay, Xiangru Huang, Artem Lukoianov, Kimia Nadjahi, David Palmer, Jack Richter-Powell, Chris Scarvelis, Tal Shnitzer, Leticia Mattos Da Silva, Mieke Moran, Dmitriy Smirnov, Oded Stein, Yu Wang, Yue Wang, Paul Zhang, and Jiacheng Zhu. I would also like to thank my friends outside my lab, in particular: Kevin Chandra, Felipe Suárez Colmenares, Gary Lee, Ning Leow, Chen Lu, Yihui Quek, Mason Rogers, Andrea Scaudo, Yan Sheng Ang, Boya Song, Dousabel Tay, Paxton Turner, Shiqi Yang, Chenyang Yuan, and Jie Yun.

A special mention goes to Morris Ang, a steadfast friend who has been a pillar in my life for the last ten years.

I also want to thank my funding sources for granting me the freedom to pursue a diverse set of research interests: the generous support of Army Research Office grants W911NF1710068 and W911NF2010168, of Air Force Office of Scientific Research award FA9550-19-1-031, of National Science Foundation grant IIS-1838071 and CHS-1955697, from the CSAIL Systems that Learn program, from the MIT–IBM Watson AI Laboratory, from the Toyota–CSAIL Joint Research Center, from a gift from Adobe Systems, from an MIT.nano Immersion Lab/NCSOFT Gaming Program seed grant, and from a Google Research Scholar award.

Contents

Chapter 1

Introduction

1.1 Distributional Optimization

Classical optimization is typically formulated as:

$$\min_{x \in \mathcal{K}} f(x), \tag{1.1}$$

where $\mathcal{K} \subset \mathbb{R}^d$ is the constraint set and $f : \mathbb{R}^d \to \mathbb{R}$ is the objective function. Significant advances have been made in the past decade towards solving (1.1). For example, if f is convex and $\mathcal{K}$ is convex, and that f is sufficiently smooth, then numerous gradient-based methods can be used to find the global optima of (1.1) with provable convergence rates.

However, the complexity of modern applications often necessitates the optimization of a *multitude* of variables. For instance, the optimization variables can be

- a swarm of particles collectively achieving a common goal [DM00; PKB07];
- a diversified set of solutions to classical or multobjective optimization [NZE05; Li+16];
- a generative model that approximates the data distribution from which infinite samples can be drawn [Pan+19; Cao+24].

To accommodate these complexities, we consider a distributional version of (1.1):

$$\min_{\mu \in \mathcal{P}(\mathcal{X})} \mathcal{F}(\mu), \tag{1.2}$$

where $\mathcal{P}(\mathcal{X})$ denotes the space of probability distributions supported on $\mathcal{X}$, and $\mathcal{F} : \mathcal{P}(\mathcal{X}) \to \mathbb{R}$ is a functional. Optionally we may include additional constraints on μ in (1.2). This formulation generalizes (1.1) by setting $\mathcal{X} = \mathcal{K}$ and $\mathcal{F}(\mu) = \int f \mathrm{d}\mu$ and noting

that the Dirac delta distribution at the minimizer of (1.1) solves (1.2). For generative modeling, we may take $\mathcal{F}(\mu) = \mathcal{D}(\mu, \mathcal{Z})$ for some probability divergence $\mathcal{D}$ and data distribution $\mathcal{Z}$, with μ typically parameterized as a neural network which imposes implicit constraints on μ.

Despite the broad applicability of distributional optimization, creating a universal framework for optimizing over distributions is difficult due to the generality of the formulation. In this thesis, we focus on specific, impactful tasks where distributional optimization proves fruitful. We propose a diverse set of scalable methodologies grounded in mathematical principles, demonstrating state-of-the-art performance in their respective domains. Tab. 1.1 summarizes the five works developed in this thesis, categorized based on how each of the following three challenges are addressed:

(a) **Parametrization.** How to represent distributions in a computationally viable way?

(b) **Modeling.** How to best define the objective function on the space of distributions?

(c) **Algorithm.** How to develop efficient optimization algorithms in theory and in practice?

Method	Parametrization	Modeling	Algorithm
Ch.2 [Li+23b]	moving particles	mollified χ^2 divergence	GD w/ reparam./barrier
Ch.3 [LDM24]	weighted coreset	MMD of zero-mean kernel	debiasing + thinning
Ch.4 [Li+20]	dual potentials	Wasserstein barycenter	SGD
Ch.5 [Li+23a]	pushforward map	proximal operator	SGD w/ global convergence
Ch.6 [LHS23]	probability flow	self-consistency condition	iterative velocity matching

Table 1.1: Overview of the included works. The horizontal line is to separate methods with discrete and neural parametrization. MMD denotes the maximum mean discrepancy (3.2), and GD (resp. SGD) stands for gradient descent (resp. stochastic gradient descent).

Although these methods are tailored to their specific applications, they exhibit characteristics that can guide future approaches to distributional optimization. Notably, they are grounded in mathematical principles, specifically in probability theory and convex optimization. Moreover, they provide clear and straightforward intuition with scalable, streamlined implementations without superfluous algorithmic components.

■ 1.2 Organization

The thesis will be divided into two parts based on whether discrete parametrization or neural parameterization is used. In the first part, the focus is on the specific task of producing high-quality samples by optimizing a collection of potentailly weighted particles.

In the second part, we exploit convexity principles and use neural networks to parameterize continuous distributions—distributions that provide infinite streams of samples—in three different tasks accompanied by scalable optmization methods.

Part I: Optimizing Interacting Particles for Producing High-Quality Samples

A fundamental problem in statistics and machine learning is sampling: generating discrete points that collectively approximate a target probability density. With a proper probability divergence as the objective, sampling is a prime example of distributional optimization where the distribution is parameterized as discrete particles, potentially with weights. In the first part of this thesis, we present two methods (Chapter 2, Chapter 3) on high-quality sampling by optimizing interaction energies of the form $\sum_{i,j=1}^{n} w_i w_j \boldsymbol{k}(x_i, x_j)$, where $(x_i)_{i=1}^n$ are positions of the samples and $(w_i)_{i=1}^n$ are weights.

Chapter 2: Generating well-separated samples with flexible constraint handling. We start by posing the design question: *How to define an objective over a collection of moving particles for the purpose of sampling?* To this end, in Chapter 2, we introduce a new interaction energy between probability measures called the mollified interaction energy (MIE). Intuitively, minimizing MIE includes an attraction force that drives samples toward a high-density region and a repulsion force that prevents samples from collapsing. By the careful construction of MIE, we show that as the mollifier approaches the Dirac delta, MIE converges to the χ^2 divergence to the target measure, and the minimizers of MIE converge to the target measure. Proper discretization of MIE yields a practical first-order particle-based algorithm for sampling in both unconstrained and constrained domains. Compared to existing algorithms like SVGD for constrained sampling, our method generates well-spreaded samples (Fig. 1.1) and handles more flexible constraints.

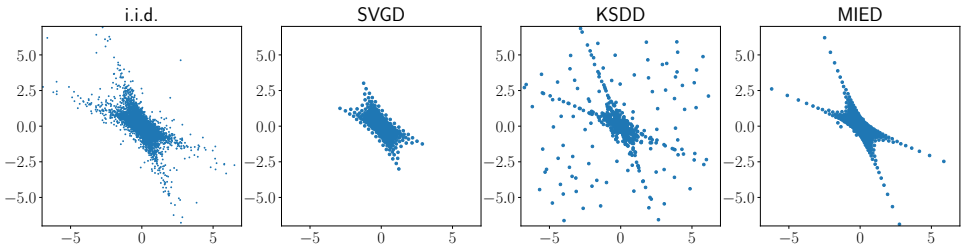

Figure 1.1: Mollified interaction energy descent (MIED) from Chapter 2 obtains well-spreaded samples from a heavy-tail distribution compared with i.i.d. samples and particle-based alternatives such as SVGD [LW16] and KSDD [Kor+21a].

Chapter 3: Compressing and debiasing millions of samples. *Given a large number of biased samples that poorly approximate the target distribution—for instance, when the Bayesian posterior distribution is challenging to sample from using Markov chain Monte Carlo (MCMC) and bias needs to be introduced for MCMC to mix well—can we distill a small coreset of high-quality samples that approximate the target distribution better?* In Chapter 3, we answer this question affirmatively by proposing a suite of algorithms with provable guarantees to simultaneously correct bias and compress the input samples given access to a mean-zero kernel (e.g. the Langevin Stein kernel). For an input of n biased samples, we introduce Stein Kernel Thinning (SKT) that produces $\sqrt{n}$ a high-quality coreset achieving the same rate of maximum mean discrepancy (MMD) as n i.i.d. unbiased samples. For larger-scale tasks, low-rank SKT achieves the same rate in sub-quadratic time. For downstream applications that support weighted coresets, we propose Stein Recombination and Stein Cholesky that match the guarantees of SKT with only poly-log(n) weighted points. Underlying these advances are new guarantees for the quality of simplex-weighted coresets, the spectral decay of kernel matrices, and the covering numbers of Stein kernel Hilbert spaces. We show empirically that our algorithms provide succinct and accurate summaries while overcoming biases due to burn-in (Fig. 1.2), approximate Markov chain Monte Carlo, and tempering.

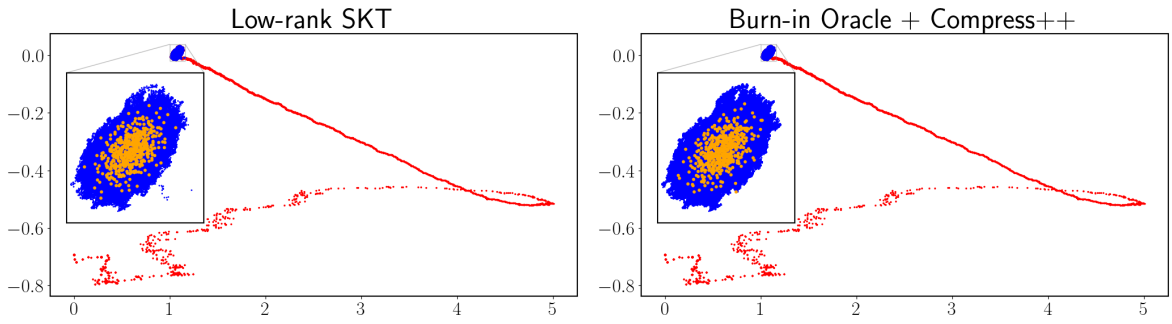

Figure 1.2: Low-rank Stein Kernel Thinning from Chapter 3 simultaneously removes the bias (red) and compresses the debiased samples (blue) to produce a high-quality coreset (orange), matching the burn-in oracle that uses 6 MCMC chains.

Part II: Optimizing Continuous Distributions with Neural Parametrization

Recent advances in deep learning suggest new ways of representing and optimizing continuous distributions parameterized using neural networks. Despite the universal approximation power of neural networks, two challenges remain:

- First, neural networks parameterize functions in the Euclidean space, but the optimization variable in distributional optimization is a probability distribution with inherent constraints (e.g. its density integrates to one) and heterogeneous desiderata (e.g. density and sample access).
- Second, unlike classical optimization, whether one can successfully optimize over neural network weights remains an active research area that depends on many factors such as the objective, the parameterization, and the optimization algorithm.

In the second part of this thesis, we demonstrate that *convexity principles* in the functional or distributional spaces can be a crucial tool to tackle these challenges, giving rise to formulations amenable with neural networks with strong performance. To this end, we present three scalable frameworks (Chapter 4, Chapter 5, Chapter 6) that optimize over continuous distributions using neural parameterization in different tasks.

Chapter 4: Computing continuous regularized Wasserstein barycenters. *How to compute the "average" of a list of continuous distributions given sample access?* Optimal transport theory provides a framework for extending Euclidean concepts of distances and averages into the realm of probability distributions. In Chapter 4, we propose an algorithm that computes the continuous Wasserstein barycenter with arbitrary ground cost, a method previously unattainable without discretization. This was accomplished by exploiting the strong duality of the regularized primal problem and parameterizing the dual potentials using neural networks combined with stochastic gradient descent. The barycenter distribution can then be recovered via the optimized transport plan (Fig. 1.3). This work has inspired several follow-up barycenter methods [Kor+21b; Kor+22] that are effective for higher dimensions.

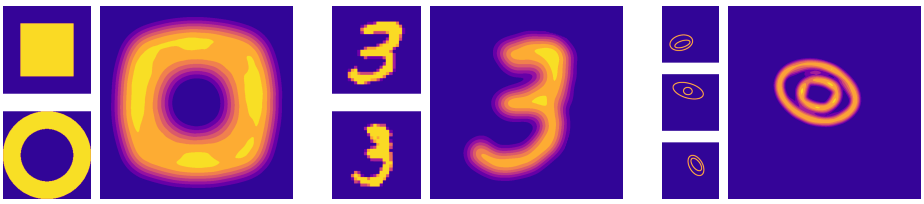

Figure 1.3: The regularized barycenter method from Chapter 4 can produce geometrically meaningful "average" distribution (right column) from the sources (left column) for each of the three examples.

Chapter 5: Finding multiple optima of classically non-convex problems. *Can we apply distributional optimization to solve classical non-convex optimization problems in*

$\mathbb{R}^d$ *and recover the all minima?* In Chapter 5, we show that lifting classical non-convex optimization to distributional optimization can be a powerful idea. We propose a framework that optimizes for the proximal operator of a family of non-convex objectives. By iterating the optimized operator parameterized as a neural network, we recover local minima as a push-forward probability distribution. The key ingredient in our formulation is a proximal regularization term, which elevates the convexity of our training loss: We show that for weakly-convex objectives with Lipschitz gradients, training of the proximal operator converges globally with a practical degree of over-parametrization. Such convergence result is previously unknown in the learning-to-optimize literature [Che+22a]. Practically, our method has obtained strong performance in a wide variety of applications including symmetry detection of 3D shapes (Fig. 1.4) and object detection in images.

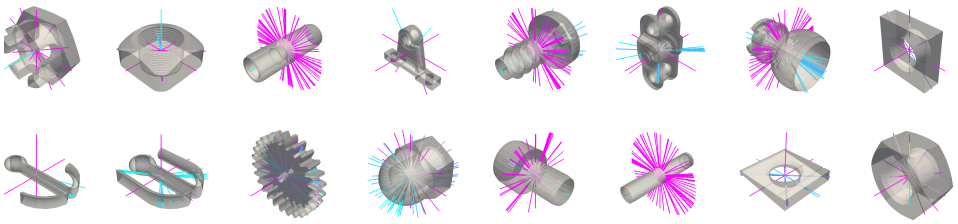

Figure 1.4: The proximal-operator-learning framework from Chapter 5 can identify an arbitrary number of reflectional symmetries of various mechanical parts in the test set. The normal of each reflectional plane is indicated by a colored line segment where pink indicates more accurate symmetry.

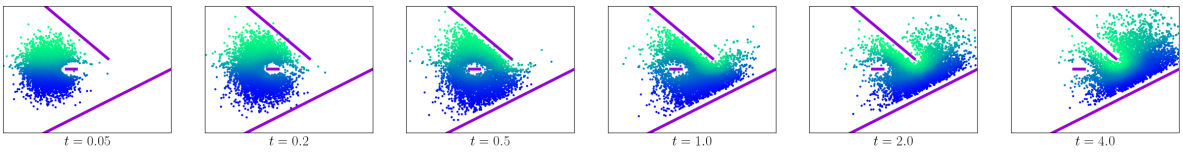

Figure 1.5: Simulation of a flow splashing against three obstacles (in purple) produced by the method from Chapter 6. Particles are colored based on the initial y coordinates.

Chapter 6: Optimizing probability flows for simulating dynamical systems. *Beyond optimizing for a single distribution, can we use optimization to recover the time evolution of a probability flow undergoing fluid dynamics or stochastic processes?* In Chapter 6, we develop a discretization-free framework for solving a large class of mass-conserving

partial differential equations (PDEs), including Fokker-Planck equations and Wasserstein gradient flows. The main observation is that the velocity field of the PDE solution needs to be self-consistent: it must satisfy a fixed-point equation that involves the probability flow determined by the current velocity field. To solve the fixed-point equation, we propose an iterative optimization scheme, reminiscent of Picard's iteration, that solves a least square problem at each iteration. This iterative scheme bypasses significant computational obstacles that one may encounter by directly minimizing the self-consistency residual. Compared to existing approaches, in addition to not suffering from temporal or spatial discretization, our method covers a wider range of PDEs including complicated dynamics (Fig. 1.5) and scales to high dimensions with less optimization time.

Chapter 7: Conclusion. In this ultimate chapter, we summarize the common themes of the methods developed that serve as a guide for future research in distributional optimization. We end the thesis with several promising future directions.

Part I

Optimizing Interacting Particles for Producing High-Quality Samples

We begin tackling the task of high-quality sampling by optimizing over discrete distributions parameterized as moving particles. By "high-quality", we mean that given the target distribution $\mathbb{P}$, the resulting points $(x_i)_{i=1}^n$ and weights $(w_i)_{i=1}^n$ together achieve a small

$$\mathcal{D}\left(\sum_{i=1}^n w_i \delta_{x_i}, \mathbb{P}\right),$$

for some probability divergence $\mathcal{D}$.

As a motivating example, in Fig. 1.6, we conduct a simple experiment of sampling uniformly within a box in $\mathbb{R}^2$, using i.i.d. sampling and the methods in Chapters 2 and 3. The i.i.d. points, while easy to generate, tend to under-sample and over-sample various regions. In contrast, our methods in Chapters 2 and 3 produce more evenly distributed points. This intuition is quantitatively verified in the right plot of Fig. 1.6, which shows that the samples generated by our methods attain higher quality in terms of the energy distance.

We focus on the case where the probability divergence $\mathcal{D}$ can be expressed as a pairwise interaction energy:

$$\mathcal{D}\left(\sum_{i=1}^n w_i \delta_{x_i}, \mathbb{P}\right) = \sum_{i,j=1}^n w_i w_j \boldsymbol{k}(x_i, x_j)$$

using a kernel $\boldsymbol{k}$, a symmetric, positive definite function $\boldsymbol{k} : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ that incorporates information about $\mathbb{P}$. Specifically:

- In Chapter 2, we utilize a mollified kernel $\boldsymbol{k}_\epsilon$ such that $\mathcal{D}$ approaches the χ^2 divergence to $\mathbb{P}$ as ϵ goes to 0, giving rise to a particle-based sampling method with flexible constraint handling.

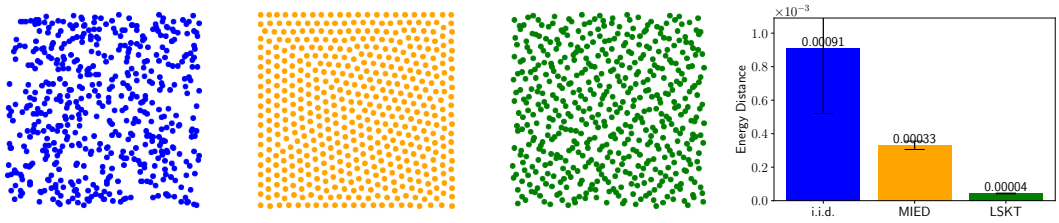

Figure 1.6: Uniform sampling in a box. Left: 512 samples generated using i.i.d. sampling (blue), mollified interaction energy descent (orange, Chapter 2), and Low-rank Stein Kernel Thinning (green, Chapter 3). Right: Energy distance [SR13] between the empirical measure formed by the 512 samples and another empirical measure formed by 65536 i.i.d. samples for each method.

- In Chapter 3, we employ a kernel $\boldsymbol{k}_{\mathbb{P}}$ that is mean-zero under $\mathbb{P}$ and $\mathcal{D}$ is the maximum mean discrepancy to $\mathbb{P}$. We develop a suite of debiased compression methods for various coreset types.

Chapter 2

Sampling via Mollified Interaction Energy Descent

In this chapter, we design a kernel $\boldsymbol{k}$, which depends on the score of $\mathbb{P}$, such that the minimizer of

$$\min_{(x_i)_{i=1}^n} \sum_{i,j=1}^n \boldsymbol{k}(x_i, x_j)$$

is a good approximation of $\mathbb{P}$. This chapter is based on the publication [Li+23b].

■ 2.1 Introduction

Sampling from an unnormalized probability density is a ubiquitous task in statistics, mathematical physics, and machine learning. While Markov chain Monte Carlo (MCMC) methods [Bro+11] provide a way to obtain unbiased samples at the price of potentially long mixing times, variational inference (VI) methods [BKM17] approximate the target measure with simpler (e.g., parametric) distributions at a lower computational cost. In this chapter, we develop a new class of VI methods that approximate the target measure using a collection of interacting particles. A prior sampling method of evolving a collection of interacting particles is Stein variational gradient descent (SVGD) proposed by [LW16], a kernelized discreization of the gradient flow of the Kullback–Leibler (KL) divergence to the target distribution.

While MCMC and VI methods have found great success in sampling from unconstrained distributions, they often break down for distributions supported in a constrained domain. Constrained sampling is needed when the target density is undefined outside a given domain (e.g., the Dirichlet distribution), when the target density is not integrable in the entire Euclidean space (e.g., the uniform distribution), or when we only want samples

that satisfy certain inequalities (e.g., fairness constraints in Bayesian inference [LTL21]). A few recent approaches [BSU12; BG13; LZ18; SLM21] extend classical sampling methods like Hamiltonian Monte Carlo (HMC) or SVGD to constrained domains. These extensions, however, typically involve expensive numerical subroutines like solving nonlinear systems of equations; moreover, explicit formulas for quantities such as Riemannian metric tensors and mirror maps need to be derived on a case-by-case basis to agree with the constraints.

We present an optimization-based method called *mollified interaction energy descent* (MIED) that minimizes *mollified interaction energies* (MIEs) for both unconstrained and constrained sampling. An MIE takes the form of a double integral of the quotient of a *mollifier*—smooth approximation of δ_0, the Dirac delta at the origin—over the target density properly scaled. Intuitively, minimizing an MIE balances two types of forces: attractive forces that drive the current measure towards the target density, and repulsive forces that prevent collapsing. We show that as the mollifier converges to δ_0, the MIE converges to the χ^2 divergence to the target measure up to an additive constant (Thm. 2.1). Moreover, the MIE Γ-converges to χ^2 divergence (Thm. 2.2), so that minimizers of MIEs converge to the target measure, providing a theoretical basis for sampling by minimization.

While mollifiers can be interpreted as kernels with diminishing bandwidths, our analysis is fundamentally different from that of SVGD where a fixed-bandwidth kernel is used to define a reproducing kernel Hilbert space (RKHS) on which the Stein discrepancy has a closed-form [GM17]. Deriving a version of the Stein discrepancy for constrained domains is far from trivial and requires special treatment [SLM21; Xu21]. In contrast, our energy has a unified form for constrained and unconstrained domains and approximates the χ^2 divergence as long as the bandwidth is sufficiently small so that short-range interaction dominates the energy: this idea of using diminishing bandwidths in sampling is under-explored for methods like SVGD.

Algorithmically, we use first-order optimization to minimize MIEs discretized using particles. We introduce a log-sum-exp trick to neutralize the effect of arbitrary scaling of the mollifiers and the target density; this form also improves numerical stability significantly. Since we turn sampling into optimization, we can readily apply existing constrained sampling techniques such as reparameterization using a differentiable (not necessarily bijective) map or the dynamic barrier method by [GL21] to handle generic differentiable inequality constraints. Our method only uses the first-order derivatives of both the target density and the inequality constraint (or the reparameterization map), enabling large-scale applications in machine learning (see e.g. Fig. 2.5). For unconstrained sampling problems, we show MIED achieves comparable performance to particle-based algorithms like SVGD, while for constrained sampling problems, MIED demonstrates strong performance compared to alternatives while being more flexible with constraint handling.

2.2 Related Works

KL gradient flow and its discretization. The Wasserstein gradient flow of the KL divergence has been extensively studied, and many popular sampling algorithms can be viewed as discretizations of the KL-divergence gradient flow. Two primary examples are Langevin Monte Carlo (LMC) and Stein variational gradient descent (SVGD). LMC simulates Langevin diffusion and can be viewed as a forward-flow splitting scheme for the KL-divergence gradient flow [Wib18]. At each iteration of LMC, particles are pulled along $-\nabla \log p$ where p is the target density, while a random Gaussian noise is injected. A Metropolis adjusting step is typically needed to counter the bias in LMC [RDF78]. In contrast with LMC, SVGD is a deterministic algorithm that updates a collection of particles using a combination of an attractive force involving $-\nabla \log p$ and a repulsive force among the particles; it can be viewed as a kernelized gradient flow of the KL divergence [Liu17] or of the χ^2 divergence [Che+20]. The connection to the continuous gradient flow in the Wasserstein space is fruitful for deriving sharp convergence guarantees for these sampling algorithms [DMM19; Bal+22; Kor+20; SSR22].

Sampling in constrained domains. Sampling in constrained domains is more challenging compared to the unconstrained setting. Typical solutions are rejection sampling and reparameterization to an unconstrained domain. However, rejection sampling can have a high rejection rate when the constrained domain is small, while reparameterization maps need to be chosen on a case-by-case basis with a determinant-of-Jacobian term that can be costly to evaluate. [BSU12] propose a constrained version of HMC for sampling on implicit submanifolds, but their algorithm is expensive as they need to solve a nonlinear system of equations for every integration step in each step of HMC. [BG13] propose geodesic Hamiltonian Monte Carlo for sampling on embedded manifolds, but they require explicit geodesic formulae. [Zha+20; AC21] propose discretizations of the mirror-Langevin diffusion for constrained sampling provided that a mirror map is given to capture the constraint. Similarly, [SLM21] propose mirror SVGD that evolves particles using SVGD-like updates in the dual space defined by a mirror map. While these two methods have theoretical convergence guarantees, they are limited by the availability of mirror maps that capture the constraints. [LTL21] extend SVGD to incorporate a population constraint over the samples obtained; their setting is different from ours: we assume the same constraint is applied to every sample.

Pairwise interaction energies on particles. Pairwise interaction energies on particles take the form $E(\{x_i\}_{i=1}^N) = \sum_{i,j} W(x_i, x_j)$ for a kernel $W(x, y)$. The gradient flow of E on N particles can give rise to phenomena like flocking and swarming, and a long line of mathematical research studies mean-field convergence of such particle gradient flow to

continuous solutions of certain PDEs as $N \to \infty$ [CCH14; Ser20]. Of particular interest to sampling, a separate line of works summarized by [BHS19] demonstrates that for the *hypersingular Riesz* kernel $W(x,y) = \|x-y\|_2^{-s}$ with sufficiently big s, minimizing E (with summation over $i \neq j$) over a compact set yields uniform samples as $N \to \infty$. Moreover, W can depend on p to obtain non-uniform samples distributed according to p. As the hypersingular Riesz kernel is not integrable, their analysis is based on geometric measure theory and bypasses variational techniques. In contrast, we take a different approach by using integrable kernels in MIEs through mollifiers. This enables us to apply variational analysis to establish interesting connections between MIEs and the χ^2 divergence. We compare our formulation with theirs further in Sec. 2.B. [Jos+19] propose to minimize the maximum of pairwise interaction akin to the interaction in [BHS19] without gradient information using a greedy algorithm. Recently, [Kor+21a] propose sampling via descending on kernel Stein discrepancy which can also be viewed as a type of interaction energy, but like SVGD, it is limited to unconstrained sampling and can be slow as higher-order derivatives of $\log p$ are required. [Cra+23] propose approximating the χ^2 divergence with a functional different from ours that also involves a mollifier. The resulting algorithm needs to evaluate an integral containing the target density over the whole domain at each step which can be costly. Consequently, their method is only applied to special target densities for which the integral has a closed form. In comparison, our method can handle general target densities with cheap updates in each step.

■ 2.3 Designing Interaction Energies

Notation. Let $X \subset \mathbb{R}^n$ be the domain we work in. We use λ_n to denote the Lebesgue measure on $\mathbb{R}^n$ and $\int f(x)\mathrm{d}x$ denotes integration with respect to λ_n. We will assume all measures are Borel. Let $\mathcal{B}_2(x,r) \triangleq \{p \in \mathbb{R}^n : \|p-x\|_2 \leq r\}$ with shorthand $\mathcal{B}_2(r) \triangleq \mathcal{B}_2(0,r)$. The d-dimensional Hausdorff measure is denoted by $\mathcal{H}_d$. We use ω_N to denote a set of points $\{x_1,\dots,x_N\} \subset \mathbb{R}^n$. For $x \in \mathbb{R}^n$, let δ_x be the Dirac delta measure at x and δ_{ω_N} be the empirical measure $\frac{1}{N}\sum_{k=1}^N \delta_{x_k}$. We denote $\mathcal{L}^p(X) \triangleq \{f : X \to \mathbb{R}$ Lebesgue measurable with $\int_X |f(x)|^p \,\mathrm{d}x < \infty\}$, and for $f \in \mathcal{L}^p(X)$ we let $\|f\|_p \triangleq (\int_X |f(x)|^p \,\mathrm{d}x)^{1/p}$. We use $\|f\|_\infty$ to denote the essential supremum of $|f|$. For $f, g : \mathbb{R}^n \to \mathbb{R}$, we write their convolution as $(f*g)(x) \triangleq \int f(y)g(x-y)\mathrm{d}y$ provided this integral exists. We use $\mathcal{C}^k(X)$ to denote continuously k-differentiable real-valued functions and $\mathcal{C}^\infty(X)$ to denote the smooth functions. The space of continuous k-differentiable functions with compact support on X is $\mathcal{C}_c^k(X)$. For vector-valued continuously k-differentiable functions from X to $Y \subset \mathbb{R}^n$, we use the notation $\mathcal{C}^k(X,Y)$ and analogously for other classes of vector-valued functions. We denote by $\mathcal{P}(X)$ the set of probability measures, and by $\mathcal{P}_2(X)$ the set of probability measures with bounded second moments. We denote by $\mathcal{M}_{\text{sign}}(X)$ the set of finite signed measures. Given a Borel measurable map $\boldsymbol{T} : X \to X$

and $\mu \in \mathcal{P}(X)$, $\boldsymbol{T}_{\#}\mu$ is the pushforward measure of μ by $\boldsymbol{T}$.

Setup. The domain $X \subset \mathbb{R}^n$ where we constrain samples is assumed to be closed and full-dimensional, i.e., $\lambda_n(X) > 0$; the unconstrained case corresponds to $X = \mathbb{R}^n$. The unnormalized target density, always denoted as p, is assumed to satisfy $p \in \mathcal{C}^1(X)$, $p(x) > 0$ for all $x \in X$, and $0 < \int_X p(x)\mathrm{d}x < \infty$. Let μ^* be the target probability measure $\mu^*(B) \triangleq \frac{\int_B p(x)\mathrm{d}x}{\int_X p(x)\mathrm{d}x}$ for a Borel set $B \subset X$. Our goal is to sample from μ^*.

♢ 2.3.1 Mollifiers

Our framework relies on the notion of *mollifiers* originally introduced in Friedrichs [Fri44].

Definition 2.1 (Family of mollifiers)**.** *We say* $\{\phi_\epsilon\}_{\epsilon>0} \subset \mathcal{C}^1(\mathbb{R}^n)$ *is a family of* mollifiers *if it satisfies:*

(a) for any $\epsilon > 0$, $x \in \mathbb{R}^n$, $\phi_\epsilon(x) \geq 0$ *and* $\phi_\epsilon(x) = \phi_\epsilon(-x)$*;*

(b) for any $\epsilon > 0$, $\|\phi_\epsilon\|_1 = 1$ *and* $\sup_{x\in\mathbb{R}^n} \phi_\epsilon(x) < \infty$*; and*

(c) for any $\delta > 0$, $p \in \{1, \infty\}$, $\lim_{\epsilon\to 0} \|\mathbf{1}_{\mathbb{R}^n \setminus \mathcal{B}_2(\delta)}\phi_\epsilon\|_p = 0$.

As $\epsilon \to 0$, the distribution with density ϕ_ϵ converges to δ_0, and if $f \in \mathcal{L}^p(\mathbb{R}^n)$ and is continuous, then the convolution $\phi_\epsilon * f$ converges to f both pointwise (Prop. 2.7) and in $\mathcal{L}^p$ (Prop. 2.8). Our definition is different from the one used in the PDE theory [Hör15], where some $\phi \in C_c^\infty(\mathbb{R}^n)$ is fixed and the family of mollifiers is generated by $\phi_\epsilon(x) \triangleq \epsilon^{-n}\phi(x/\epsilon)$. While in the PDE theory mollifiers are typically used to improve the regularity of non-smooth functions, in our framework they are used to construct interaction energies that approximate the χ^2 divergence.

We will use the following mollifiers. In the ensuing definitions, we include normalizing constants Z_ϵ that ensure $\|\phi_\epsilon\|_1 = 1$. We do not need the explicit form of Z_ϵ as such constants do not appear in our practical algorithm (Alg. 2.1). For $s > n$, the *s-Riesz family of mollifiers* is defined as $\phi_\epsilon^s(x) \triangleq (\|x\|_2^2 + \epsilon^2)^{-s/2}/Z_\epsilon^s$. The *Gaussian family of mollifiers* is defined as $\phi_\epsilon^g(x) \triangleq \exp\left(-\frac{\|x\|_2^2}{2\epsilon^2}\right)/Z_\epsilon^g$. The *Laplace family of mollifiers* is defined as $\phi_\epsilon^l(x) \triangleq \exp\left(-\frac{\|x\|_2}{\epsilon}\right)/Z_\epsilon^l$. Since $\{\phi_\epsilon^g\}$ and $\{\phi_\epsilon^l\}$ correspond to the densities of centered Gaussian and Laplace random variables, they satisfy Def. 2.1. Prop. 2.6 proves that the s-Riesz family $\{\phi_\epsilon^s\}$ also satisfies Def. 2.1.

♢ 2.3.2 Mollified interaction energies

Definition 2.2 (Mollified interaction energy)**.** *Given a family of mollifiers* $\{\phi_\epsilon\}_{\epsilon>0}$ *(Def. 2.1), for each* $\epsilon > 0$*, define a symmetric kernel* $W_\epsilon : \mathbb{R}^n \times \mathbb{R}^n \to [0, \infty]$ *by*

$$W_\epsilon(x, y) \triangleq \begin{cases} \phi_\epsilon(x - y)(p(x)p(y))^{-1/2} & \text{if } x, y \in X \\ \infty & \text{otherwise.} \end{cases} \tag{2.1}$$

Define the mollified interaction energy *(MIE),* $\mathcal{E}_\epsilon : \mathcal{P}(\mathbb{R}^n) \to [0, \infty]$*, to be*

$$\mathcal{E}_\epsilon(\mu) \triangleq \iint W_\epsilon(x, y)\mathrm{d}\mu(x)\mathrm{d}\mu(y). \tag{2.2}$$

Intuitively, minimizing $\mathcal{E}_\epsilon(\mu)$ balances two forces: a repulsive force from $\phi_\epsilon(x - y)$ makes sure μ has a good spread, and an attractive force from $(p(x)p(y))^{-1/2}$ helps μ concentrate on high-density regions. The exponent $-1/2$ is chosen to balance the two forces so that, as we will show in Thm. 2.1, $\mathcal{E}_\epsilon(\mu)$ approximates the χ^2 divergence with respect to μ^* for small ϵ.

♢ 2.3.3 Convergence to χ^2 divergence

Recall that the χ^2-divergence between probability measures $\mathbb{P}, \mathbb{Q}$ is defined by

$$\chi^2(\mathbb{Q} \parallel \mathbb{P}) \triangleq \begin{cases} \int \left(\frac{\mathrm{d}\mathbb{Q}}{\mathrm{d}\mathbb{P}} - 1\right)^2 \mathrm{d}\mathbb{P} & \text{if } \mathbb{Q} \ll \mathbb{P} \\ \infty & \text{otherwise,} \end{cases}$$

where $\mathbb{Q} \ll \mathbb{P}$ denotes absolute continuity of $\mathbb{Q}$ with respect to $\mathbb{P}$ with density $\frac{\mathrm{d}\mathbb{Q}}{\mathrm{d}\mathbb{P}}$.

We start by giving an alternative formula for χ^2 divergence with respect to the target measure.

Lemma 2.1. *Define a functional* $\mathcal{E} : \mathcal{P}(\mathbb{R}^n) \to [0, \infty]$ *by*

$$\mathcal{E}(\mu) \triangleq \begin{cases} \int_X \frac{q(x)^2}{p(x)}\mathrm{d}x & \text{if } \mathrm{d}\mu(x) = q(x)\mathrm{d}\lambda_n(x) \text{ and } \mu \ll \mu^* \\ \infty & \text{otherwise.} \end{cases}$$

Then for every $\mu \in \mathcal{P}(\mathbb{R}^n)$*,*

$$\mathcal{E}(\mu) = \chi^2(\mu \parallel \mu^*) + 1.$$

Proof. If μ is not absolutely continuous with respect to μ^*, then both sides are infinity. Otherwise, $\mu(X) = 1$ and μ has some density q. We then compute

$$\begin{aligned} \chi^2(\mu \parallel \mu^*) + 1 &= \int \left(\frac{\mathrm{d}\mu}{\mathrm{d}\mu^*}(x) - 1\right)^2 \mathrm{d}\mu^*(x) + 1 = \int_X \left(\frac{q(x)}{p(x)} - 1\right)^2 p(x)\mathrm{d}x + 1 \\ &= \int_X \frac{q(x)^2}{p(x)}\mathrm{d}x - 2\int_X q(x)\mathrm{d}x + 2 = \int_X \frac{q(x)^2}{p(x)}\mathrm{d}x = \mathcal{E}(\mu). \end{aligned}$$

□

Theorem 2.1 (Pointwise convergence to χ^2 divergence)**.** *Suppose $\mu \in \mathcal{P}(\mathbb{R}^n)$ satisfies $\chi^2(\mu \parallel \mu^*) < \infty$. Then, $\mathcal{E}_\epsilon(\mu) < \infty$ for any $\epsilon > 0$. Furthermore,*

$$\lim_{\epsilon \to 0} \mathcal{E}_\epsilon(\mu) = \chi^2(\mu \parallel \mu^*) + 1.$$

Below we give a succinct proof of Thm. 2.1 using the theory of mollifiers developed in Sec. 2.A.1.

Proof of Thm. 2.1. Since $\chi^2(\mu \parallel \mu^*) < \infty$, we know $\mu \ll \mu^*$, so $\mu(X) = 1$. We let $q(x)$ be the density of μ which satisfies $q(x) = 0$ for $x \in \mathbb{R}^n \setminus X$. We compute using Tonelli's theorem (since our integrand is positive):

$$\begin{aligned}\mathcal{E}_\epsilon(\mu) &= \iint \phi_\epsilon(x-y)(p(x)p(y))^{-1/2} q(x)q(y)\mathrm{d}x\mathrm{d}y \\ &= \int \left(\phi_\epsilon * \frac{q}{\sqrt{p}}\right)(x) \frac{q}{\sqrt{p}}(x)\mathrm{d}x = \int (\phi_\epsilon * g)(x) g(x)\mathrm{d}x, \end{aligned} \tag{2.3}$$

where we define $g(x) \triangleq q(x)/\sqrt{p(x)}$ on X and $g(x) \triangleq 0$ for $x \notin X$. Moreover, by Lem. 2.1,

$$\chi^2(\mu \parallel \mu^*) + 1 = \int g(x)^2 \mathrm{d}x.$$

The assumption $\chi^2(\mu \parallel \mu^*) < \infty$ then implies $g \in \mathcal{L}^2(\mathbb{R}^n)$. By Prop. 2.8, we have $\phi_\epsilon * g \in \mathcal{L}^2(\mathbb{R}^n)$. Next notice that

$$\left|\mathcal{E}_\epsilon(\mu) - \left(\chi^2(\mu \parallel \mu^*) + 1\right)\right| \le \int |(\phi_\epsilon * g)(x) - g(x)|\, g(x)\mathrm{d}x \le \|\phi_\epsilon * g - g\|_2 \|g\|_2 < \infty.$$

This shows the first claim. Finally, the last expression goes to 0 as $\epsilon \to 0$ since $\|\phi_\epsilon * g - g\|_2 \to 0$ by Prop. 2.8. □

Remark 2.1. *One may ask if similar results as Thm. 2.1 can be obtained for f-divergences other than the χ^2 divergence. We highlight that the proof of Thm. 2.1 is highly specific to the χ^2 divergence because in (2.3) there are two copies of q coming from the definition of $\mathcal{E}_\epsilon$ as a double integral over μ. Any f-divergence between μ and μ^* has the form $\int_X f(q(x)/p(x))\mathrm{d}p(x)$, and the only way to make q^2 appear is to choose $f(x) = x^2$ which gives rise to the χ^2 divergence.*

Remark 2.2. *It is possible to prove versions of Thm. 2.1 when X has Hausdorff dimension $d < n$: in such cases the χ^2-divergence still makes sense and so does (2.2). When X is "flat", i.e., with Hausdorff dimension d and contained in a d-dimensional linear subspace of $\mathbb{R}^n$, e.g., when X is defined by a set of linear inequalities, then Thm. 2.1 follows if we adapt the assumptions of Def. 2.1 and calculations in the proof of Thm. 2.1*

to be in the subspace. For a general d-dimensional X, the same calculation yields $\mathcal{E}_\epsilon(\mu) = \int_X \left(\int_X \phi_\epsilon(x-y)\left(\frac{q}{\sqrt{p}}\right)(y)\mathrm{d}\mathcal{H}_d(y)\right)\left(\frac{q}{\sqrt{p}}\right)(x)\mathrm{d}\mathcal{H}_d(x)$. For a similar argument to go through, we will need the normalizing constant $\int_X \phi_\epsilon(x-y)\mathrm{d}\mathcal{H}_d(y)$ to the same for all x. This is true for example when X is a d-dimensional sphere, but for a general X the last integral will depend on the base point x. Proving a version of Thm. 2.1 for a generic X with Hausdorff dimension $d < n$ is an interesting future direction.

♢ 2.3.4 Convexity and Γ-convergence

We next study properties of the minimizers of MIEs: Does $\min_{\mu\in\mathcal{P}(X)} \mathcal{E}_\epsilon(\mu)$ admit a unique minimum? If so, do minima of $\{\mathcal{E}_\epsilon\}_{\epsilon>0}$ converge to μ^* as $\epsilon \to 0$? In order to answer affirmatively to these questions, we will need the associated kernel $\boldsymbol{k}_\epsilon(x,y) \triangleq \phi_\epsilon(x-y)$ to satisfy the following property.

Definition 2.3 (I.s.p.d. kernel)**.** *A symmetric lower semicontinuous (l.s.c.) kernel $\boldsymbol{k}$ on $X \times X$ is* integrally strictly positive definite *(i.s.p.d.) if for every finite signed Borel measure ν on X, the energy $\mathcal{E}_{\boldsymbol{k}}(\nu) \triangleq \iint \boldsymbol{k}(x,y)\mathrm{d}(\nu\times\nu)(x,y)$ is well-defined (i.e., the integrals over the negative and positive parts of ν are not both infinite), and $\mathcal{E}_{\boldsymbol{k}}(\nu) \geq 0$ where the equality holds only if $\nu = 0$ on all Borel sets of X.*

For the mollifiers we consider, the associated kernel $\boldsymbol{k}_\epsilon(x,y) = \phi_\epsilon(x-y)$ is i.s.p.d. on any compact set (Pronzato and Zhigljavsky [PZ21, Example 1.2]).

The next result, proved in Sec. 2.A.2, shows W_ϵ is i.s.p.d. and $\mathcal{E}_\epsilon$ attains a unique minimum whenever $\boldsymbol{k}_\epsilon$ is i.s.p.d. and X is compact.

Proposition 2.1 (Uniqueness of the minimizer of $\mathcal{E}_\epsilon$)**.** *Suppose X is compact and $\boldsymbol{k}_\epsilon(x,y) \triangleq \phi_\epsilon(x-y)$ is i.s.p.d. on X. Then,*

(a) the kernel $W_\epsilon(x,y)$ defined in (2.1) also i.s.p.d on X, and

(b) the functional $\mathcal{E}_\epsilon$ is strictly convex on $\mathcal{M}_{\text{sign}}(X)$ and attains a unique minimum on $\mathcal{P}(X)$.

The rest of the section is dedicated to showing the convergence of minima of $\{\mathcal{E}_\epsilon\}$ to μ^*. This follows as a consequence of Γ-convergence of the sequence $\{\mathcal{E}_\epsilon\}$, defined as follows.

Definition 2.4 (Γ-convergence)**.** *A sequence of functionals $\mathcal{F}_\epsilon : \mathcal{P}(\mathbb{R}^n) \to (-\infty,\infty]$ is said to* Γ-converge *to $\mathcal{F} : \mathcal{P}(\mathbb{R}^n) \to (-\infty,\infty]$, denoted as $\mathcal{F}_\epsilon \xrightarrow{\Gamma} \mathcal{F}$, if:*

(a) for any sequence $\mu_\epsilon \in \mathcal{P}(\mathbb{R}^n)$ converging weakly to $\mu \in \mathcal{P}(\mathbb{R}^n)$, $\liminf_{\epsilon\to 0} \mathcal{F}_\epsilon(\mu_\epsilon) \geq \mathcal{F}(\mu)$, and

(b) for any $\mu \in \mathcal{P}(\mathbb{R}^n)$, there exists a sequence $\mu_\epsilon \in \mathcal{P}(\mathbb{R}^n)$ converging weakly to μ with $\limsup_{\epsilon\to 0} \mathcal{F}_\epsilon(\mu_\epsilon) \le \mathcal{F}(\mu)$.

Theorem 2.2 (Γ-convergence to χ^2 divergence)**.** *Suppose $\boldsymbol{k}_\epsilon(x,y) \triangleq \phi_\epsilon(x-y)$ is i.s.p.d. on compact sets for every $\epsilon > 0$. Then we have Γ-convergence (Def. 2.4) $\mathcal{E}_\epsilon \xrightarrow{\Gamma} \chi^2(\cdot \parallel \mu^*) + 1$ as $\epsilon \to 0$. In particular, if X is compact, if we denote $\mu_\epsilon^* \triangleq \arg\min_{\mu\in\mathcal{P}(X)} \mathcal{E}_\epsilon(\mu)$, then $\mu_\epsilon^* \to \mu^*$ weakly and $\lim_{\epsilon\to 0} \mathcal{E}_\epsilon(\mu_\epsilon^*) = 1$.*

Thm. 2.2 provides basis for minimizing $\mathcal{E}_\epsilon$ for a small ϵ, since its unique minimum will be a good approximation of μ^*.

The main tool towards proving Thm. 2.2 is the following result developed in Sec. 2.A.2, which give an expression of the double integral of an i.s.p.d. kernel using its Fourier transform (Def. 2.6):

Proposition 2.2. *Suppose $\phi \in \mathcal{L}^1(\mathbb{R}^n)$ is even, bounded, continuous, and $\boldsymbol{k}(x,y) \triangleq \phi(x-y)$ is i.s.p.d. on any compact sets. Let $\hat{\phi}$ denote the Fourier transform of ϕ. Then, for any $\nu \in \mathcal{M}_{\text{sign}}(\mathbb{R}^n)$, if we let $\hat{\nu}$ denote the Fourier transform of ν, it holds that*

$$\iint \phi(x-y)\mathrm{d}\nu(x)\mathrm{d}\nu(y) = \int |\hat{\nu}(\xi)|^2\, \hat{\phi}(\xi)\mathrm{d}\xi.$$

Proof of Thm. 2.2. First note that Def. 2.4(b) is immediate from Thm. 2.1 with $\mu_\epsilon = \mu$: if $\mathcal{E}(\mu) = \infty$, then trivially $\limsup_{\epsilon\to 0} \mathcal{E}_\epsilon(\mu) \le \mathcal{E}(\mu)$; otherwise we apply Thm. 2.1. So we focus on proving criterion Def. 2.4(a).

Fix a sequence $\mu_\epsilon \in \mathcal{P}(\mathbb{R}^n)$ that converges weakly to $\mu \in \mathcal{P}(\mathbb{R}^n)$, and our goal is to show $\liminf_{\epsilon\to 0} \mathcal{E}_\epsilon(\mu_\epsilon) \ge \mathcal{E}(\mu)$. Without loss of generality, we may assume $\mathcal{E}_\epsilon(\mu_\epsilon) < \infty$ for all $\epsilon > 0$ (these terms have no effect in $\liminf$), which implies $\mu_\epsilon(X) = 1$. By Portmanteau's theorem and the assumption that X is closed, we have $\mu(X) \ge \limsup_{\epsilon\to 0} \mu_\epsilon(X) = 1$. So all of μ_ϵ and μ will have support in X.

For $m > 0, \epsilon > 0$, define a nonnegative measure $\nu_{\epsilon,m}$ by

$$\mathrm{d}\nu_{\epsilon,m}(x) \triangleq h_m(x) p^{-1/2}(x) \mathrm{d}\mu_\epsilon(x),$$

where $h_m : \mathbb{R}^n \to \mathbb{R}$ is a continuous monotonically decreasing cutoff function satisfying $h_m(x) = 1$ if $\|x\|_2^2 < m$ and $h_m(x) = 0$ if $\|x\|_2^2 > m+1$, and that $h_m(x) \le h_{m'}(x)$ for $m < m'$. Then since $p > 0$ is continuous, it is bounded below on any compact set in X, and hence $\nu_{\epsilon,m}$ is finite. Also define, for $m > 0$, a nonnegative measure ν_m by

$$\mathrm{d}\nu_m(x) \triangleq h_m(x) p^{-1/2}(x) \mathrm{d}\mu(x),$$

which is again finite following the same reasoning.

Then for any $m > 0$, denoting $\mathrm{d}\nu_\epsilon(x) \triangleq p^{-1/2}(x)\mathrm{d}\mu_\epsilon(x)$,

$$\begin{aligned}\mathcal{E}_\epsilon(\mu_\epsilon) &= \iint \phi_\epsilon(x-y)\mathrm{d}\nu_\epsilon(x)\mathrm{d}\nu_\epsilon(y) \\ &\geq \iint \phi_\epsilon(x-y)\mathrm{d}\nu_{\epsilon,m}(x)\mathrm{d}\nu_{\epsilon,m}(y) = \int |\hat{\nu}_{\epsilon,m}(\xi)|^2 \hat{\phi}_\epsilon(\xi)\mathrm{d}\xi, \end{aligned} \tag{2.4}$$

where we apply Prop. 2.2 for the last equality. On the other hand, note that

$$\hat{\nu}_{\epsilon,m}(\xi) = \int e^{-2\pi i\xi\cdot x}\mathrm{d}\nu_{\epsilon,m}(x) = \int e^{-2\pi i\xi\cdot x} h_m(x)p^{-1/2}(x)\mathrm{d}\mu_\epsilon(x).$$

Since $\mu_\epsilon \to \mu$ weakly and the last integrand is a continuous bounded function, we have

$$\lim_{\epsilon\to 0} \hat{\nu}_{\epsilon,m}(\xi) = \int e^{-2\pi i\xi\cdot x} h_m(x)p^{-1/2}(x)\mathrm{d}\mu(x) = \hat{\nu}_m(\xi).$$

On the other hand, $\hat{\phi}_\epsilon(\xi) = \int e^{-2\pi i\xi\cdot x}\phi_\epsilon(x)\mathrm{d}x$. Since by Def. 2.1, $\phi_\epsilon \mathrm{d}x$ converges to δ_0 in probability (and in particular weakly), we have $\lim_{\epsilon\to 0}\hat{\phi}_\epsilon(\xi) = 1$.

Applying Fatou's lemma to (2.4), we obtain, for any $m > 0$,

$$\liminf_{\epsilon\to 0} \mathcal{E}_\epsilon(\mu_\epsilon) \geq \int |\hat{\nu}_m(\xi)|^2 \,\mathrm{d}\xi.$$

If $\int |\hat{\nu}_m(\xi)|^2 \,\mathrm{d}\xi = \infty$ for any $m > 0$, then we are done since $\liminf_{\epsilon\to 0}\mathcal{E}_\epsilon(\mu_\epsilon) = \infty$. Otherwise, by Kühn [Küh16, Lemma 2.11], for every $m > 0$, ν_m has density in $\mathcal{L}^2(\mathbb{R}^n)$. This imples μ has density everywhere. Suppose $\mathrm{d}\mu(x) = q(x)\mathrm{d}\lambda_n(x)$. By Plancherel's theorem and the monotone convergence theorem, we have

$$\begin{aligned}\liminf_{\epsilon\to 0} \mathcal{E}_\epsilon(\mu_\epsilon) &\geq \lim_{m\to\infty} \int |\hat{\nu}_m(\xi)|^2 \,\mathrm{d}\xi \\ &= \lim_{m\to\infty} \int_X \left(h_m(x)p^{-1/2}(x)q(x)\right)^2 \mathrm{d}x \\ &= \int_X p^{-1}(x)q(x)^2\mathrm{d}x = \mathcal{E}(\mu).\end{aligned}$$

This completes the proof of $\mathcal{E}_\epsilon \xrightarrow{\Gamma} \mathcal{E}$.

Now suppose X is compact, and let $\mu_\epsilon^* \triangleq \arg\min_{\mu\in\mathcal{P}(X)} \mathcal{E}_\epsilon(\mu)$. To establish $\mu_\epsilon^* \to \mu^*$ weakly (resp. $\lim_{\epsilon\to 0}\mathcal{E}_\epsilon(\mu_\epsilon^*) = \mathcal{E}(\mu^*) = 1$), it suffices to show that every subsequence of $\{\mu_\epsilon^*\}$ (resp. $\{\mathcal{E}_\epsilon(\mu_\epsilon^*)\}$) has a further convergence subsequence converging to μ^* (resp. $\mathcal{E}(\mu^*)$). With a slight abuse of notation, we assume the sequence of ϵ is chosen so that $\{\mu_\epsilon^*\}$ (resp. $\{\mathcal{E}_\epsilon(\mu_\epsilon^*)\}$) is already some subsequence of the original sequence. As argued in the proof of Lem. 2.5, $\mathcal{P}(X)$ is compact with respect to weak convergence. Hence $\{\mu_\epsilon^*\}$

has a weakly convergence subsequence $\mu^*_{\epsilon_k} \to \nu$ for some $\nu \in \mathcal{P}(X)$. The Γ-convergence $\mathcal{E}_\epsilon \xrightarrow{\Gamma} \mathcal{E}$ implies

$$\liminf_{k\to\infty} \mathcal{E}_{\epsilon_k}(\mu^*) \geq \liminf_{k\to\infty} \mathcal{E}_{\epsilon_k}(\mu^*_{\epsilon_k}) \geq \mathcal{E}(\nu) \geq \limsup_{k\to\infty} \mathcal{E}_{\epsilon_k}(\nu) \geq \limsup_{k\to\infty} \mathcal{E}_{\epsilon_k}(\mu^*_{\epsilon_k}),$$

where we have used, for each inequality, $\mu^*_{\epsilon_k} = \arg\min_{\mathcal{P}(X)} \mathcal{E}_{\epsilon_k}$, Def. 2.4(a), the first paragraph of this proof, and again the fact that $\mu^*_{\epsilon_k} = \arg\min_{\mathcal{P}(X)} \mathcal{E}_{\epsilon_k}$. Since $\liminf_{k\to\infty} \mathcal{E}_{\epsilon_k}(\mu^*) = \mathcal{E}(\mu^*)$ by Thm. 2.1, we have

$$\mathcal{E}(\mu^*) \geq \lim_{k\to\infty} \mathcal{E}_{\epsilon_k}(\mu^*_{\epsilon_k}) = \mathcal{E}(\nu) \geq \mathcal{E}(\mu^*),$$

where the last inequality follows because μ^* is the minimizer of $\mathcal{E}$. Then $\lim_{k\to\infty} \mathcal{E}_{\epsilon_k}(\mu^*_{\epsilon_k}) = \mathcal{E}(\mu^*) = 1$. Moreover, $\chi^2(\nu \parallel \mu^*) = 0$. This can only happen if ν and μ^* agree on all Borel sets, so $\nu = \mu^*$. □

♢ 2.3.5 Differential calculus of $\mathcal{E}_\epsilon$ in $\mathcal{P}_2(\mathbb{R}^n)$

We next study the gradient flow of $\mathcal{E}_\epsilon$ in Wasserstein space $\mathcal{P}_2(\mathbb{R}^n)$ for $X = \mathbb{R}^n$. Understanding the gradient flow of a functional often provides insights into the convergence of algorithms that simulates gradient flow with time discretization [AGS05, Chapter 11] or spatial discretization [Chi22]. Proofs of this section are given in Sec. 2.A.3.

Let $\mathcal{F} : \mathcal{P}_2(\mathbb{R}^n) \to \mathbb{R}$ be a functional. The *Wasserstein gradient flow* of $\mathcal{F}$ [AGS05, Definition 11.1.1] is defined as the solution $\{\mu_t\}_{t\geq 0}$ of the PDE: $\frac{\partial \mu_t}{\partial t} = \nabla \cdot (\mu_t \boldsymbol{w}_{\mathcal{F},\mu_t})$ where $\boldsymbol{w}_{\mathcal{F},\mu} \in \mathcal{L}^2(\mu; \mathbb{R}^n)$ is a *Frechét subdifferential* (Def. 2.7) of $\mathcal{F}$ at μ. Intuitively, gradient flows capture the evolution of the variable being optimized if we were to do gradient descent[1] on $\mathcal{F}$ when the step sizes go to 0. We next show that the gradient flow of $\mathcal{E}_\epsilon$ agrees with that of $\chi^2(\cdot \parallel \mu^*)$ in the sense that their subdifferentials coincide as $\epsilon \to 0$.

Proposition 2.3. *Assume $\mu \in \mathcal{P}_2(\mathbb{R}^n)$ has density $q \in C^1_c(\mathbb{R}^n)$[2]. Then any strong Frechét subdifferential (Def. 2.7) of $\mathcal{E}_\epsilon$ at μ takes the form*

$$\boldsymbol{w}_{\epsilon,\mu}(x) = 2\nabla \left(p(x)^{-1/2} (\phi_\epsilon * q/\sqrt{p})(x) \right), \text{ for } \mu\text{-a.e. } x \in \mathbb{R}^n. \tag{2.5}$$

Moreover, if for sufficiently small ϵ, ϕ_ϵ has compact support independent of ϵ, then

$$\lim_{\epsilon\to 0} \boldsymbol{w}_{\epsilon,\mu}(x) = \boldsymbol{w}_{\chi^2,\mu}(x), \text{ for } \mu\text{-a.e. } x \in \mathbb{R}^n, \tag{2.6}$$

where $\boldsymbol{w}_{\chi^2,\mu}$ is a strong Frechét subdifferential of $\chi^2(\cdot \parallel \mu^*)$.

[1] The more precise notion is of *minimizing movements* [AGS05, Chapter 2].

[2] We assume that μ has compact support because W_ϵ can be unbounded and cause integrability issues due to the presence of $(p(x)p(y))^{-1/2}$ term.

While simulating χ^2 divergence gradient flow is often intractable [TS20, Section 3.3], our MIE formulation admits a practical algorithm (Sec. 2.4) without requiring the density of the flow.

We next show that $\mathcal{E}_\epsilon$ is *displacement convex* at $\mu^* \in \mathcal{P}_2(\mathbb{R}^n)$ as $\epsilon \to 0$, obtaining a result similar to Korba et al. [Kor+21a, Corollary 4]. This suggests that gradient flow initialized near μ^* will have nice convergence guarantee.

Proposition 2.4. *Suppose $p \in \mathcal{C}_c^2(\mathbb{R}^n)$. Suppose that for sufficiently small ϵ, ϕ_ϵ has compact support independent of ϵ. Assume $\boldsymbol{k}_\epsilon(x,y) \triangleq \phi_\epsilon(x-y)$ is i.s.p.d. Let $\boldsymbol{\xi} \in C_c^\infty(\mathbb{R}^n, \mathbb{R}^n)$ and $\mu_t \triangleq (\boldsymbol{I} + t\boldsymbol{\xi})_\# \mu^*$. Then $\lim_{\epsilon\to 0} \frac{\mathrm{d}^2}{\mathrm{d}t^2}\Big|_{t=0} \mathcal{E}_\epsilon(\mu_t) \geq 0$.*

We now consider a time discretization of the gradient flow of $\mathcal{E}_\epsilon$ given by, for an initial measure $\mu_0 \in \mathcal{P}_2(\mathbb{R}^n)$ and $m \in \mathbb{N}_0$, with a step size $\gamma > 0$,

$$\mu_{m+1} \triangleq (\boldsymbol{I} - \gamma \boldsymbol{w}_{\epsilon,\mu_m})_\# \mu_m. \tag{2.7}$$

Following Korba et al. [Kor+21a, Proposition 14], we can show a descent lemma for iterations (2.7) provided the following smoothness assumptions:

Assumption 2.1. *Suppose the target density $p \in \mathcal{C}^2(\mathbb{R}^n)$ satisfies $1/C_p \leq p(x) \leq C_p$, $\|p(x)\|_2 \leq C_p'$, $\|\mathrm{H}\, p(x)\|_2 \leq C_p''$ for some C_p, C_p', C_p'' for all $x \in \mathbb{R}^n$, where we use $\|A\|_2$ to indicate the matrix spectral norm (i.e. $\|A\|_2 = \sigma_{\max}(A)$). Suppose the mollifier $\phi_\epsilon \in \mathcal{C}^2(\mathbb{R}^n)$ satisfies, in addition to Def. 2.1, $\phi_\epsilon(x) \leq C_\epsilon$, $\|\nabla \phi_\epsilon(x)\|_2 \leq C_\epsilon'$, and $\|\mathrm{H}\, \phi_\epsilon(x)\|_2 \leq C_\epsilon''$ for some C_ϵ, C_ϵ', C_ϵ''.*

Proposition 2.5 (Descent lemma for $\mathcal{E}_\epsilon$)**.** *Under Assum. 2.1, suppose $\mu_0 \in \mathcal{P}_2(\mathbb{R}^n)$ has compact support. Then for any $\gamma \leq \frac{1}{2L}$, with*

$$L \triangleq C_\epsilon'' C_r^2 + 2C_\epsilon' C_r' C_r + C_\epsilon \max(C_r'' C_r, C_r'^2), \tag{2.8}$$

the update (2.7) satisfies, for each $m \in \mathbb{N}_0$,

$$\mathcal{E}_\epsilon(\mu_{m+1}) - \mathcal{E}_\epsilon(\mu_m) \leq -\gamma(1 - 2\gamma L)\|\boldsymbol{w}_{\epsilon,\mu_m}\|_{\mathcal{L}^2(\mu_m)} \leq 0. \tag{2.9}$$

■ 2.4 Mollified Interaction Energy Descent

We now present a first-order particle-based algorithm for constrained and unconstrained sampling by minimizing a discrete version of (2.2). Substituting an empirical distribution for μ into (2.2) gives the discrete mollified interaction energy, for N particles $\omega_N = \{x_1, \ldots, x_N\}$,

$$E_\epsilon(\omega_N) \triangleq \frac{1}{N^2} \sum_{i=1}^N \sum_{j=1}^N \phi_\epsilon(x_i - x_j)(p(x_i)p(x_j))^{-1/2}. \tag{2.10}$$

Algorithm 2.1 Mollified interaction energy descent (MIED) in the logarithmic domain.

Input: target density p, mollifier ϕ_ϵ, initial particles $\{x_i^0\}_{i=1}^N$, total steps T, gradient update function `GradientUpdate`
for $t \leftarrow 1$ **to** T **do**
  for $i \leftarrow 1$ **to** N **do**
    ▷ $I_{ij} \triangleq \log \phi_\epsilon(x_i - x_j) - \frac{1}{2}(\log p(x_i) + \log p(x_j))$; *see the derivation in Sec. 2.C*
    $-\nabla_{x_i} \log E_\epsilon \leftarrow \sum_{j=1}^N \frac{e^{I_{ij}}}{\sum_{i,j} e^{I_{ij}}} (2\nabla \log \phi_\epsilon(x_j - x_i) + \nabla \log p(x_i))$
    $x_i^{t+1} \leftarrow$ `GradientUpdate` $(x_i^t, -\nabla_{x_i} \log E_\epsilon)$
  end for
end for
Return: final particles $\{x_i^T\}_{i=1}^N$

Denote $\omega_{N,\epsilon}^* \in \arg\min_{\omega_N \subset X} E_\epsilon(\omega_N)$ and $\mu_\epsilon^* = \arg\min_{\mu \in \mathcal{P}_2(X)} \mathcal{E}_\epsilon(\mu)$, If X is compact, by Borodachov, Hardin, and Saff [BHS19, Corollary 4.2.9], we have weak convergence $\delta_{\omega_{N,\epsilon}^*} \to \mu_\epsilon^*$ as $N \to \infty$. If in addition $k_\epsilon(x, y) \triangleq \phi_\epsilon(x - y)$ is i.s.p.d., then by Thm. 2.2, we have weak convergence $\mu_\epsilon^* \to \mu^*$. This shows that minimizing (2.10) with a large N and a small ϵ will result in an empirical distribution of particles that approximates μ^*. Our sampling method, mollified interaction energy descent (MIED) described in Alg. 2.1, is simply minimizing (2.10) using gradient-based algorithms. For instance, for unconstrained sampling, we obtain an instantiation of MIED using gradient descent (Alg. 2.1) with an update that resembles the one in SVGD (see the discussion in Sec. 2.C.1). Below we address a few practical concerns.

Optimization in the logarithmic domain. In practical applications, we only have access to the unnormalized target density p. The normalizing constant of the mollifier ϕ_ϵ can also be hard to compute (e.g., for the Riesz family). While these normalization constants do not affect the minima of (2.10), they can still affect gradient step sizes during optimization. Moreover, ϕ_ϵ can be very large when ϵ is small, and in many Bayesian applications p can be tiny and only $\log p$ is numerically significant. To address these issues, we optimize the logarithm of (2.10) using the log-sum-exp trick [BHH21] to improve numerical stability and to get rid of the arbitrary scaling of the normalizing constants:

$$\log E_\epsilon(\omega_N) \triangleq \log \sum_{i=1}^N \sum_{j=1}^N \exp(\log \phi_\epsilon(x_i - x_j) - \frac{1}{2}(\log p(x_i) + \log p(x_j))) - 2 \log N. \quad (2.11)$$

Special treatment of the diagonal terms. Since ϕ_ϵ goes to the Dirac delta as $\epsilon \to 0$, the discretization of (2.2) on a neighborhood of the diagonal $\{(x, y) : x = y\}$ needs to be handled with extra care. In (2.10) the diagonal appears as $\sum_{i=1}^N \phi_\epsilon(0) p(x_i)^{-1}$ which

can dominate the summation when ϵ is small and then $\phi_\epsilon(0)$ becomes too large. For the mollifiers that we consider, we use a different diagonal term $\sum_{i=1}^N \phi_\epsilon(h_i/\kappa_n)p(x_i)^{-1}$ where $h_i \triangleq \min_{j\neq i}\|x_i - x_j\|$ and $\kappa_n \geq 1$ is a constant depending only on the dimension n. Since $0 \leq \phi_\epsilon(h_i/\kappa_n) \leq \phi_\epsilon(0)$, the energy obtained will be bounded between (2.10) and the version of (2.10) without the diagonal terms. Hence by the proof of Theorem 4.2.2 of Borodachov, Hardin, and Saff [BHS19], the discrete minimizers of E_ϵ still converge to μ_ϵ^* as $N \to \infty$ and $\epsilon \to 0$. Empirically we found the choice $\kappa_n = (1.3n)^{1/n}$ works well for the Riesz family of mollifiers and we use the Riesz family primarily for our experiments.

Constraint handling. If $X \neq \mathbb{R}^n$, we need to minimize (2.11) subject to constraints $\omega_N = \{x_1, \ldots, x_N\} \subset X$. Since the constraint is the same for each particle x_i, we want our algorithm to remain parallelizable across particles.

We consider two types of constraints: (a) there exists a differentiable map $f : \mathbb{R}^n \to X$, with $\lambda_n(X \setminus f(\mathbb{R}^n)) = 0$; (b) the set X is given by $\{x \in \mathbb{R}^n : g(x) \leq 0\}$ for a differentiable $g : \mathbb{R}^n \to \mathbb{R}$. For (a), we reduce the problem to unconstrained optimization in $\mathbb{R}^n$ using objective $\log E_\epsilon(f(\omega_N))$ with $\omega_N = \{x_1, \ldots, x_N\} \subset \mathbb{R}^n$ and $f(\omega_N) \triangleq \{f(x_1), \ldots, f(x_N)\}$. For (b), we apply the dynamic barrier method by Gong and Liu [GL21] to particles in parallel. In Sec. 2.C.2, we extend the dynamic barrier method to handle multiple constraints.

■ 2.5 Experiments

We compare MIED with recent alternatives on unconstrained and constrained sampling problems. Unless mentioned otherwise, we choose the s-Riesz family of mollifiers $\{\phi_\epsilon^s\}$ with $s = n + 10^{-4}$ and $\epsilon = 10^{-8}$: we found minimizing the MIE with such mollifiers results in well-separated particles so that we can take ϵ to be very small as our theory recommends. This is not the case for the Gaussian or the Laplace family as setting ϵ too small can cause numerical issues even when particles are well-separated. In Sec. 2.D.5, we compare different choices of s on a constrained mixture distribution.

Unless mentioned otherwise: for SVGD [LW16], we use the Gaussian kernel with adaptive bandwidths as in the original implementation. While it is rarely discussed in the literature, we found such adaptive rules of SVGD can be prone to collapse samples—see Sec. 2.D.2; for KSDD [Kor+21a], we use a fixed Gaussian kernel with unit variance. All methods by default use a learning rate of 0.01 with Adam optimizer [KB14]. The source code can be found at https://github.com/lingxiaoli94/MIED.

♢ 2.5.1 Unconstrained sampling

Gaussians in varying dimensions. We first compare MIED with SVGD and KSDD on Gaussians of varying dimensions and with different numbers of particles. In Fig. 2.1, we

see that as the number of dimensions increases, MIED results in best samples in terms of W_2 distance (Wasserstein-2 distance computed using linear programming without entropic regularization) with respect to 10^4 i.i.d. samples, while SVGD yields lower energy distance [SR13]. We think this is because MIED results in more evenly spaced samples (Fig. 2.D.1) so the W_2 distance is lower. More details can be found in Sec. 2.D.1.

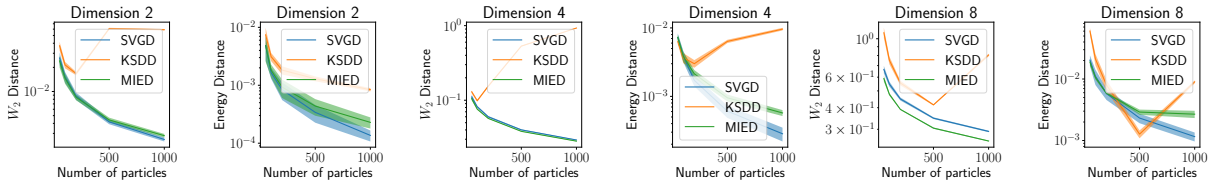

Figure 2.1: Gaussian experiments results in varying dimensions and different numbers of particles. For each method, we plot the metric (W_2 distance or energy distance) versus the number of particles, averaged over 10 trials (shaded region indicates standard deviation).

Product of two Student's t-distributions. Next, we consider a 2-dimensional distribution constructed as the product of two independent t-distributions with the degree of freedom $\nu = 2$ composed with a linear transform. Unlike Gaussians, Student's t-distributions have heavy tails. On the left of Fig. 2.2, we visualize the results of each method with 1000 samples after 2000 iterations. MIED captures the heavy tail while SVGD fails. Quantitatively, on the right of Fig. 2.2, while SVGD captures the distribution in $[-a, a]^2$ better for $a \leq 3$, MIED yields lower metrics for a bigger a.

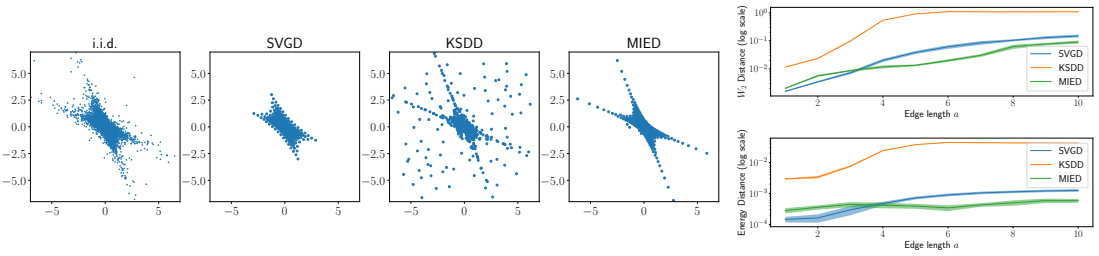

Figure 2.2: Left: visualization of samples from each method for the product of two Student's t-distribution (composed with a linear transform). Right: metrics of each method when restricting to $[-a, a]^2$. As t-distributions have heavy tails, if we draw i.i.d. samples from the true product distribution, a small number of them will have large norms, making the computation of metrics unstable. Thus we restrict both i.i.d. samples and the resulting samples from each method to $[-a, a]^2$ before computing the metrics for $a \in [2, 10]$.

Bayesian logistic regression. We compare MIED with SVGD and KSDD for the Bayesian logistic regression setting in Liu and Wang [LW16]. We further include a naïve baseline, independent particle descent (IPD), which simply runs gradient descent independently on each particle to maximize the posterior probability. In addition to using test accuracy as the metric, we include W_2 distance and energy distance between the samples from each method and 10^4 samples from NUTS [HG+14] after sufficient burn-in steps. We summarize the results in Tab. 2.1. All methods, including the naïve baseline IPD, are comparable in terms of test accuracy. In other words, accuracy is not a good metric for comparing the quality of posterior samples. Bayesian inference is typically preferred over maximum a posteriori estimation for its ability to capture uncertainty. When evaluating the quality of the uncertainty of the samples using distributional distances, MIED provides the best approximation in terms of the W_2 distance and all methods (except IPD) are comparable in terms of the energy distance.

Dataset (d)	NUTS	IPD	SVGD	KSDD	MIED
banana (3)	0.55	-6.14/-3.58/**0.55**	-7.81/-5.24/**0.55**	**-8.24**/**-5.76**/**0.55**	-7.37/-5.06/**0.55**
breast cancer (10)	0.64	-1.51/-1.03/**0.60**	-1.62/-2.06/**0.60**	-1.71/**-2.23**/**0.60**	**-1.99**/-2.18/**0.60**
diabetis (9)	0.78	-2.18/-1.55/**0.77**	-3.09/-3.42/**0.77**	-2.91/**-3.90**/**0.77**	**-3.11**/-3.13/**0.77**
flare solar (10)	0.59	3.30/2.65/0.48	6.91/4.09/0.52	**1.77**/**-0.08**/**0.55**	7.09/4.25/0.48
german (21)	0.65	-1.80/-1.25/**0.65**	-1.89/-2.63/0.64	-1.27/**-2.83**/**0.65**	**-1.96**/-2.80/**0.65**
heart (14)	0.87	-0.40/-0.56/**0.87**	-0.41/-1.50/**0.87**	-0.10/**-1.76**/**0.87**	**-0.92**/-1.67/**0.87**
image (19)	0.82	6.53/4.31/**0.83**	7.17/4.01/**0.83**	2.16/-0.50/0.82	**1.14**/**-1.88**/0.82
ringnorm (21)	0.77	-3.82/-2.45/**0.77**	**-4.11**/**-5.98**/**0.77**	1.07/-2.21/0.76	-4.03/-5.70/**0.77**
splice (61)	0.85	**-1.47**/-1.18/**0.85**	-1.22/**-2.65**/**0.85**	2.04/-0.05/0.84	1.45/0.70/0.82
thyroid (6)	0.84	1.95/0.53/**0.84**	1.17/-0.00/**0.84**	2.42/1.57/0.74	**0.84**/**-0.37**/**0.84**
titanic (4)	0.40	**-1.59**/-0.16/**0.40**	-0.46/-0.31/**0.40**	-0.63/-0.39/**0.40**	-1.00/**-0.45**/**0.40**
twonorm (21)	0.97	-1.21/-1.13/**0.97**	-1.32/-2.78/**0.97**	1.55/-0.62/**0.97**	**-1.44**/**-3.21**/**0.97**
waveform (22)	0.77	-2.67/-1.87/**0.78**	-2.98/**-5.23**/**0.78**	-2.60/-4.18/0.77	**-3.09**/-3.17/**0.78**

Table 2.1: Bayesian logistic regression results with 1000 particles. We include the test accuracy for NUTS in the second column. Three numbers A/B/C in the following columns are logarithmic W_2 distance, logarithmic energy distance, and test accuracy. Bold indicates the best numbers. We use 80%/20% training/test split. All methods are run with identical initialization and learning rate 0.01. Results are reported after 10^4 iterations.

♢ 2.5.2 Constrained sampling

Uniform sampling in 2D. We consider uniform sampling in the square $[-1, 1]^2$ with 500 particles. We reparameterize our particles using tanh to eliminate the constraint and show results with various choices of mollifiers—we always choose the smallest ϵ while the optimization remains stable. We compare our method with mirror LMC [Zha+20] and

SVMD/MSVGD by Shi, Liu, and Mackey [SLM21] using the entropic mirror map

$$\phi(\theta) = \sum_{i=1}^{n} \left((1+\theta_i)\log(1+\theta_i) + (1-\theta_i)\log(1-\theta_i)\right).$$

To demonstrate that SVGD and KSDD break down in constrained domains, we implement these two methods adapted to the constrained setting using the same reparameterization as our method. The initial particles are drawn uniformly from $[-0.5, 0.5]^2$.

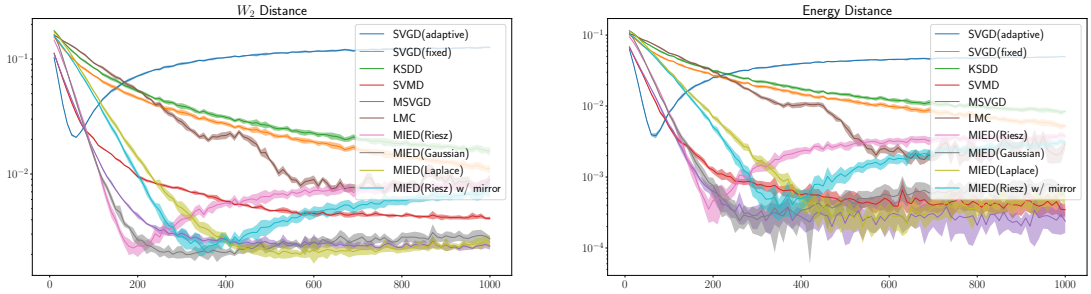

Figure 2.3: Convergence of metrics for uniform sampling from a 2D box.

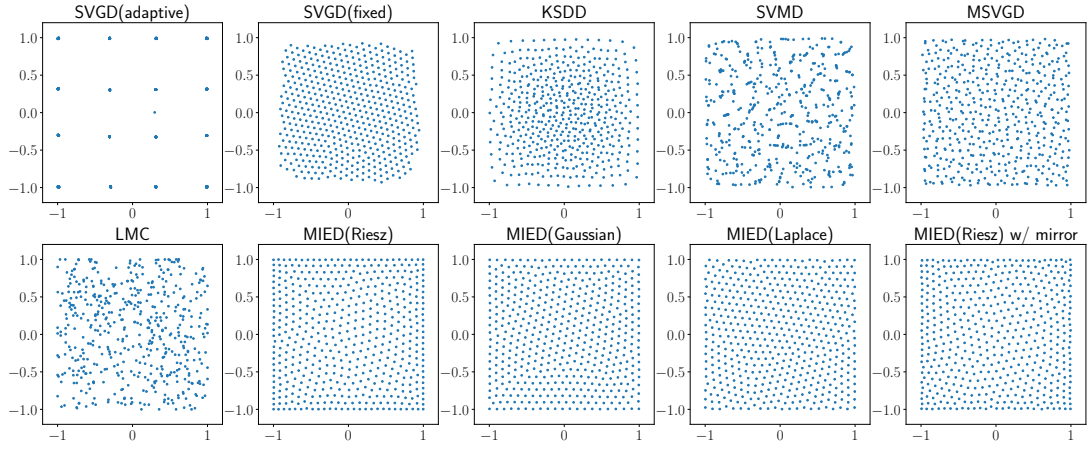

Figure 2.4: Visualization of samples at iteration 1000 for uniform sampling from a 2D box.

Fig. 2.3 shows that quantitatively our method achieves much lower energy distance and W_2 distance (measured against 5000 i.i.d. samples uniformly drawn in $[-1, 1]^2$.) compared to SVGD and KSDD with reparameterization. Of all methods, MIED with Gaussian of Laplace mollifiers and MSVGD achieve the lowest metrics. In Fig. 2.4, we visualize samples from each method. We see that samples from SVGD with an adaptive kernel

collapse—we investigate this issue in Sec. 2.D.2. Samples from SVGD with a fixed kernel size and KSDD are non-uniform (we choose kernel sizes that produce the best result). While SVMD creates locally clustered artifacts, MSVGD produces good results. For mirror LMC with the same mirror map as in MSVGD, the resulting samples are not evenly spaced, resulting in worse W_2 distances. When using tanh as the reparameterization map, MIED produces decent results for Gaussian and Laplace mollifiers; the samples obtained with the Riesz mollifier are slightly worse as samples are too dense along the edge of the square. When using the same entropic mirror map for reparameterization as in MSVGD, MIED produces comparable results as using tanh. This highlights the flexibility of our method with constraint handling, whereas, for LMC and SVMD/MSVGD, the mirror map has to be chosen carefully to be the gradient of a strongly convex function while capturing the constraints.

In Sec. 2.D.3, we further test SVMD/MSVGD with a different choice of the mirror map where they break down. In Sec. 2.D.4, we conduct a similar comparison for sampling from a 20-dimensional Dirichlet distribution using the same setup as Shi, Liu, and Mackey [SLM21]. In scenarios where a good choice of a mirror map is available, SVMD/MSVGD can obtain better performance compared to MIED. In Sec. 2.D.6, we conduct additional qualitative experiments for MIED, demonstrating its effectiveness for challenging constraints and multi-modal distributions.

Fairness Bayesian neural networks. We train fair Bayesian neural networks to predict whether the annual income of a person is at least $\$50,000$ with gender as the protected attribute using the *Adult Income* dataset [Koh+96]. We follow the same setup as in Liu, Tong, and Liu [LTL21] where the dataset $\mathcal{D} = \{x^{(i)}, y^{(i)}, z^{(i)}\}_{i=1}^{|\mathcal{D}|}$ consists of feature vectors $x^{(i)}$, labels $y^{(i)}$ (whether the income is $\geq \$50,000$), and genders $z^{(i)}$ (protected attribute). The target density is taken to be the posterior of logistic regression with a two-layer Bayesian neural network $\hat{y}(\cdot;\theta)$ with weights θ, and we put a standard Gaussian prior on each entry of θ independently. Given $t > 0$, the fairness constraint is $g(\theta) = (\mathrm{Cov}_{(x,y,z)\sim\mathcal{D}}[z, \hat{y}(x;\theta)])^2 - t \leq 0$. On the left of Fig. 2.5, we plot the trade-off curve of the result obtained using our method and the methods from Liu, Tong, and Liu [LTL21] for $t \in \{10^{-5}, 10^{-4}, 0.0001, 0.001, 0.002, 0.005, 0.01\}$. Details can be found in Sec. 2.D.7. Our method recovers a much larger Pareto front compared to the alternatives. On the right of Fig. 2.5, we visualize the curves of the energy and the covariance versus the number of training iterations: as expected we see a smaller t results in bigger MIE (lower log-likelihood) and smaller covariance between the prediction and the protected attribute.

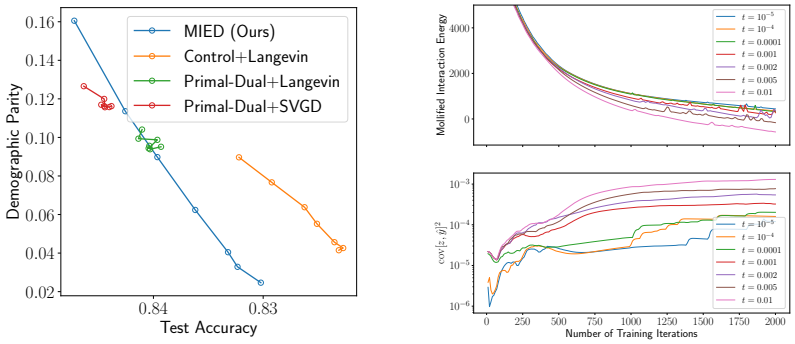

Figure 2.5: Left: trade-off curves of demographic parity versus accuracy on the test data for MIED and methods from Liu, Tong, and Liu [LTL21]. Right: MIEs and $(\mathrm{Cov}_{(x,y,z)\sim\mathcal{D}}[z, \hat{y}(x;\theta)])^2$ (measured on the training data) versus the number of training iterations, for various t.

■ 2.6 Conclusion and Future Directions

We present a new sampling method by minimizing MIEs discretized as particles for unconstrained and constrained sampling. This is motivated by the insight that MIEs converge to χ^2 divergence with respect to the target measure as the mollifier parameter goes to 0. The proposed method achieves promising results on the sampling problems we consider.

Below we highlight three limitations. First, as discussed in Rem. 2.2, our theory only applies when the domain is full-dimensional or flat. Extending our theory to handle cases where the domain is an arbitrary d-rectifiable set is an important next step as it allows the handling of more complicated constraints such as nonlinear equality constraints. Secondly, when $\epsilon > 0$, the minimizer of MIE can be different from the target measure. Finding a way to debias MIE (e.g., like how Sinkhorn distances are debiased [Fey+19]) is an interesting direction. Lastly, the connection between the minimizers of the discretized MIE (2.10) and those of the continuous MIE (2.2) is only established in the limit as $N \to \infty$. We hope to investigate how well the empirical distribution of particles minimizing (2.10) approximates the target measure when N is finite as in [XKS22].

Appendices

■ 2.A Deferred Analysis Details

♢ 2.A.1 Preliminaries on mollifiers

We review the theory of mollifiers in this section.

Proposition 2.6. *For $s > n$, the s-Riesz family of mollifiers, defined as $\phi_\epsilon^s(x) \triangleq (\|x\|_2^2 + \epsilon^2)^{-s/2}/Z_\epsilon^s$, satisfies Def. 2.1.*

In order to prove Prop. 2.6, we first prove a lemma.

Lemma 2.2. *Let $b \geq 0$. Then for any $\epsilon > 0$,*

$$Z_\epsilon^s \int_{\mathcal{B}_2(\epsilon)} \phi_\epsilon^s(y)\|y\|_2^b \mathrm{d}y = \mathcal{H}_{n-1}(S^{n-1}) \left(\int_0^1 \frac{t^{n+b-1}}{(t^2+1)^{s/2}} \right) \epsilon^{n+b-s}, \tag{2.12}$$

where $\mathcal{H}_{n-1}(S^{n-1})$ is the volume of the $(n-1)$-dimensional sphere.

Furthermore, assuming $s > b + n$, then for any $\epsilon > 0, \delta > 0$,

$$Z_\epsilon^s \int_{\mathbb{R}^n \setminus \mathcal{B}_2(\delta)} \phi_\epsilon^s(y)\|y\|_2^b \mathrm{d}y \leq \mathcal{H}_{n-1}(S^{n-1}) \frac{\delta^{n+b-s}}{s-(n+b)}. \tag{2.13}$$

Proof. For (2.12), we compute

$$\begin{aligned} Z_\epsilon^s \int_{\mathcal{B}_2(\epsilon)} \phi_\epsilon^s(y)\|y\|_2^b \mathrm{d}y &= \int_{\mathcal{B}_2(\epsilon)} \frac{\|y\|_2^b}{(\|y\|_2^2+\epsilon^2)^{s/2}} = \mathcal{H}_{n-1}(S^{n-1}) \int_0^\epsilon \frac{r^{n+b-1}}{(r^2+\epsilon^2)^{s/2}} \mathrm{d}r \\ &= \mathcal{H}_{n-1}(S^{n-1}) \int_0^1 \frac{(\epsilon t)^{n+b-1}}{(\epsilon^2 t^2+\epsilon^2)^{s/2}} \epsilon \mathrm{d}t \\ &= \mathcal{H}_{n-1}(S^{n-1}) \left(\int_0^1 \frac{t^{n+b-1}}{(t^2+1)^{s/2}} \mathrm{d}t \right) \epsilon^{n+b-s}, \end{aligned}$$

where we use substitution $r = \epsilon t$ on the second line.

If $s > b + n$, then

$$\begin{aligned} Z_\epsilon^s \int_{\mathbb{R}^n \setminus \mathcal{B}_2(\delta)} \phi_\epsilon^s(0,y)\|y\|_2^b \mathrm{d}y &= \int_{\mathbb{R}^n \setminus \mathcal{B}_2(\delta)} \frac{\|y\|_2^b}{(\|y\|_2^2 + \epsilon^2)^{s/2}} \\ &\le \int_{\mathbb{R}^n \setminus \mathcal{B}_2(\delta)} \frac{\|y\|_2^b}{\|y\|_2^s} = \mathcal{H}_{n-1}(S^{n-1}) \int_\delta^\infty r^{n-1+b-s} \\ &= \mathcal{H}_{n-1}(S^{n-1}) \frac{\delta^{n+b-s}}{s-(n+b)}, \end{aligned}$$

where the last integral uses $s > b + n$. □

Proof of Prop. 2.6. It is clear that (a) of Prop. 2.6 holds for $\phi_\epsilon^s(x)$. Since $\phi_\epsilon^s(x)$ is bounded we have $\phi_\epsilon^s \in \mathcal{L}^\infty(\mathbb{R}^n)$. For any $\delta > 0$, we have, for $\epsilon \le \delta$,

$$Z_\epsilon^s \int_{\mathcal{B}_2(\delta)} \phi_\epsilon^s(x)\mathrm{d}x \ge Z_\epsilon^s \int_{\mathcal{B}_2(\epsilon)} \phi_\epsilon^s(x)\mathrm{d}x = C\epsilon^{n-s},$$

where we use (2.12) with $b = 0$ and C is a constant depending only on n, s. On the other hand, using (2.13) with $b = 0$,

$$Z_\epsilon^s \int_{\mathbb{R}^n \setminus \mathcal{B}_2(\delta)} \phi_\epsilon^s(x)\mathrm{d}x \le C'\delta^{n-s},$$

where C' depends only on n, s. With $\delta = \epsilon$, we see that $Z_\epsilon^s \phi_\epsilon^s \in \mathcal{L}^1(\mathbb{R}^n)$ so (b) is satisfied.

Since δ is fixed and $s > n$, we see that

$$\lim_{\epsilon \to 0} \frac{\int_{\mathcal{B}_2(\delta)} \phi_\epsilon^s(x)\mathrm{d}x}{\int_{\mathbb{R}^n \setminus \mathcal{B}_2(\delta)} \phi_\epsilon^s(x)\mathrm{d}x} \ge \lim_{\epsilon \to 0} \frac{C\epsilon^{n-s}}{C'\delta^{n-s}} = \infty.$$

Since $\int_{\mathcal{B}_2(\delta)} \phi_\epsilon^s(x)\mathrm{d}x + \int_{\mathbb{R}^n \setminus \mathcal{B}_2(\delta)} \phi_\epsilon^s(x)\mathrm{d}x = 1$, we have shown (c). □

In the rest of this section, we assume $\{\phi_\epsilon\}_{\epsilon > 0}$ is a family of mollifiers satisfying Def. 2.1.

Lemma 2.3. *For any $p \in [1, \infty]$, for any $\delta > 0$, $\phi_\epsilon \in \mathcal{L}^p(\mathbb{R}^n)$ and $\lim_{\epsilon \to 0} \|\mathbf{1}_{\mathbb{R}^n \setminus \mathcal{B}_2(\delta)} \phi_\epsilon\|_p = 0$. holds.*

Proof. Assume $p \notin \{1, \infty\}$ since both cases are covered in the assumptions. Then for any $\delta > 0$, by Hölder's inequality,

$$\|\phi_\epsilon\|_p^p = \|\phi_\epsilon \cdot \phi_\epsilon^{p-1}\|_1 \le \|\phi_\epsilon\|_1 \|\phi_\epsilon^{p-1}\|_\infty = \|\phi_\epsilon\|_1 \|\phi_\epsilon\|_\infty^{p-1} < \infty.$$

Similarly,

$$\begin{aligned} \|\mathbf{1}_{\mathbb{R}^n \setminus \mathcal{B}_2(\delta)} \phi_\epsilon\|_p^p &= \|\mathbf{1}_{\mathbb{R}^n \setminus \mathcal{B}_2(\delta)} \phi_\epsilon^p\|_1 = \|\phi_\epsilon \cdot \mathbf{1}_{\mathbb{R}^n \setminus \mathcal{B}_2(\delta)} \phi_\epsilon^{p-1}\|_1 \\ &\le \|\phi_\epsilon\|_1 \|\mathbf{1}_{\mathbb{R}^n \setminus \mathcal{B}_2(\delta)} \phi_\epsilon^{p-1}\|_\infty = \|\mathbf{1}_{\mathbb{R}^n \setminus \mathcal{B}_2(\delta)} \phi_\epsilon\|_\infty^{p-1}. \end{aligned}$$

Letting $\epsilon \to 0$ and applying (c) gives $\lim_{\epsilon \to 0} \|\mathbf{1}_{\mathbb{R}^n \setminus \mathcal{B}_2(\delta)} \phi_\epsilon\|_p = 0$. □

Proposition 2.7. *Let $f \in \mathcal{L}^p(\mathbb{R}^n)$, $p \in [1, \infty]$. Then for every $\epsilon > 0$, the integral*

$$\int f(x-y)\phi_\epsilon(y)\mathrm{d}y$$

exists, so that $(f * \phi_\epsilon)(x)$ is finite. Moreover, if f is continous at x, then

$$\lim_{\epsilon \to 0}(f * \phi_\epsilon)(x) = f(x). \tag{2.14}$$

Proof. Observe that by Hölder's inequality, for q such that $1/p + 1/q = 1$ (allowing infinity),

$$\int |f(x-y)\phi_\epsilon(y)|\,\mathrm{d}y \le \|f\|_p \|\phi_\epsilon\|_q < \infty.$$

Hence $(f * \phi_\epsilon)(x)$ is integrable and finite.

For any $\epsilon > 0$, note that

$$|(f * \phi_\epsilon)(x) - f(x)| = \left| \int f(x-y)\phi_\epsilon(y)\mathrm{d}y - f(x) \right|.$$

Since $\int \phi_\epsilon(x)\mathrm{d}x = 1$, we have

$$|(f * \phi_\epsilon)(x) - f(x)| = \left| \int (f(x-y) - f(x))\,\phi_\epsilon(y)\mathrm{d}y \right| \le \int |f(x-y) - f(x)|\,\phi_\epsilon(y)\mathrm{d}y.$$

Fix $t > 0$. Continuity of f at x implies there exists $\delta > 0$ such that $|f(x-y) - f(x)| < t$ for all $y \in \mathcal{B}_2(\delta)$. Then

$$\int_{\mathcal{B}_2(\delta)} |f(x-y) - f(x)|\,\phi_\epsilon(y)\mathrm{d}y \le t \int_{\mathcal{B}_2(\delta)} \phi_\epsilon(y)\mathrm{d}y \le t.$$

On the other hand, by Hölder's inequality,

$$\int_{\mathbb{R}^n \setminus \mathcal{B}_2(x,\delta)} |f(x-y) - f(x)|\,\phi_\epsilon(y)\mathrm{d}y \le \|\tau_x f - f\|_p \|\mathbf{1}_{\mathbb{R}^n \setminus \mathcal{B}_2(\delta)}\phi_\epsilon\|_q \le 2\|f\|_p \|\mathbf{1}_{\mathbb{R}^n \setminus \mathcal{B}_2(\delta)}\phi_\epsilon\|_q.$$

Hence

$$|(f * \phi_\epsilon)(x) - f(x)| \le t + 2\|f\|_p \|\mathbf{1}_{\mathbb{R}^n \setminus \mathcal{B}_2(\delta)}\phi_\epsilon\|_q.$$

By Lem. 2.3, since $\|f\|_p < \infty$, taking $\epsilon \to 0$ we get, for any $t > 0$,

$$\limsup_{\epsilon \to 0} |(f * \phi_\epsilon)(x) - f(x)| \le t.$$

Now let $t \to 0$ we get (2.14). □

Corollary 2.1. *Let $f \in \mathcal{L}^p(\mathbb{R}^n)$, $p \in [1, \infty]$, be a continuous function. If $\{f_\epsilon\}$ is a sequence of functions that converge uniformly to f, then for every x,*

$$\lim_{\epsilon \to 0} (f_\epsilon * \phi_\epsilon)(x) = f(x).$$

Proof. Note that

$$|(\phi_\epsilon * f_\epsilon)(x) - f(x)| \leq |(\phi_\epsilon * f_\epsilon)(x) - (\phi_\epsilon * f)(x)| + |(\phi_\epsilon * f)(x) - f(x)|.$$

Prop. 2.7 shows the second term goes to 0 as $\epsilon \to 0$. For the first term, we have

$$\begin{aligned} |(\phi_\epsilon * f_\epsilon)(x) - (\phi_\epsilon * f)(x)| &= \left| \int (f_\epsilon(x-y) - f(x-y))\, \phi_\epsilon(y) \mathrm{d}y \right| \\ &\leq \sup_x |f_\epsilon(x) - f(x)| \to 0 \end{aligned}$$

by uniform convergence. □

Lemma 2.4. *For $f \in \mathcal{L}^p(\mathbb{R}^n)$, $p \in [1, \infty)$, we have*

$$\lim_{y \to 0} \|\tau_y f - f\|_p = 0,$$

where we use $\tau_a f$ to denote the translated function $\tau_a f(x) \triangleq f(x - a)$.

Proof. Fix $\epsilon > 0$. It is a standard fact that $C_c(\mathbb{R}^n)$ is dense in $\mathcal{L}^p(\mathbb{R}^n)$. Hence there exists $g \in C_c(\mathbb{R}^n)$ such that $\|f - g\|_p < \epsilon$. Since g is continuous with compact support, it is uniform continuous. Then there exists $\delta > 0$ with $|g(x-y) - g(x)| < \epsilon^{1/p}/\lambda_n(K)$ if $y \in \mathcal{B}_2(x, \delta)$. Hence for such y we have $\|\tau_y g - g\|_p^p < \epsilon$ with $\lambda_n(K) < \infty$. Thus

$$\|\tau_y f - f\|_p \leq \|\tau_y f - \tau_y g\|_p + \|\tau_y g - g\|_p + \|g - f\|_p \leq 3\epsilon.$$

□

Proposition 2.8. *Let $f \in \mathcal{L}^p(\mathbb{R}^n)$, $p \in (1, \infty)$. Then for every $\epsilon > 0$,*

$$\|f * \phi_\epsilon\|_p \leq 3\|f\|_p.$$

In particular, $f * \phi_\epsilon \in \mathcal{L}^p(\mathbb{R}^n)$. Moreover,

$$\lim_{\epsilon \to 0} \|f * \phi_\epsilon - f\|_p = 0.$$

Proof. For every $\epsilon > 0$, we have

$$\begin{aligned}\|f * \phi_\epsilon - f\|_p^p &= \int \left| \int (f(x-y) - f(x))\phi_\epsilon(y)\mathrm{d}y \right|^p \mathrm{d}x \\ &\le \int \int |f(x-y) - f(x)|^p \, \phi_\epsilon(y)\mathrm{d}y\mathrm{d}x \\ &= \int \left(\int |f(x-y) - f(x)|^p \, \mathrm{d}x \right) \phi_\epsilon(y)\mathrm{d}y \\ &= \int \|\tau_y f - f\|_p^p \phi_\epsilon(y)\mathrm{d}y,\end{aligned}$$

where we use Jensen's inequality with the observation that $\phi_\epsilon(y)\mathrm{d}y$ is a probability measure and Tonelli's theorem to exchange the order of integration. To show the first claim, note that,

$$\|f * \phi_\epsilon - f\|_p^p \le \int \|\tau_y f - f\|_p^p \phi_\epsilon(y)\mathrm{d}y \le \int (2\|f\|_p)^p \phi_\epsilon(y)\mathrm{d}y = (2\|f\|_p)^p.$$

Hence $\|f * \phi_\epsilon\|_p \le \|f\|_p + \|f * \phi_\epsilon - f\|_p \le 3\|f\|_p$.

For the second claim, by Lem. 2.4, the function $y \mapsto \|\tau_y f - f\|_p^p$ is continuous at $y = 0$ with limit 0. Hence by Prop. 2.7, we are done by taking $\epsilon \to 0$. □

♢ 2.A.2 Convexity and Γ-convergence

We start by recalling a few definitions regarding functionals in $\mathcal{P}(\mathbb{R}^n)$.

Definition 2.5. *We say a functional $\mathcal{F} : \mathcal{P}(\mathbb{R}^n) \to (-\infty, \infty]$ is* proper *if there exists $\mu \in \mathcal{P}(\mathbb{R}^n)$ such that $\mathcal{F}(\mu) < \infty$, and is* lower semicontinuous *(l.s.c.) if for any weakly convergence sequence $\mu_k \to \mu$, we have $\liminf_{k\to\infty} \mathcal{F}(\mu_k) \ge \mathcal{F}(\mu)$.*

Lemma 2.5. *For any $\epsilon > 0$, the functional $\mathcal{E}_\epsilon : \mathcal{P}(\mathbb{R}^n) \to (-\infty, \infty]$ is proper and l.s.c. Moreover, if X is compact, the minimum $\min_{\mu\in\mathcal{P}(X)} \mathcal{E}_\epsilon(\mu)$ is attained by some measure in $\mathcal{P}(X)$.*

Proof. Taking any $x \in X$, since ϕ_ϵ is bounded and $p(x) > 0$, we see that $\mathcal{E}_\epsilon(\delta_x) < \infty$ so $\mathcal{E}_\epsilon$ is proper. Moreover, given weakly convergence $\mu_k \to \mu$, by Portmanteau theorem and the fact that W_ϵ is nonnegative and l.s.c., we conclude that $\mathcal{E}_\epsilon$ is also l.s.c.

The set of probability distributions $\mathcal{P}(X) \subset \mathcal{P}(\mathbb{R}^n)$ is tight by the compactness of X. It is closed since if $\{\mu_k\} \subset \mathcal{P}(X)$ weakly converges to μ, then by Portmanteau theorem, $\mu(X) \ge \limsup \mu(X) = 1$ so that $\mu \in \mathcal{P}(X)$. Hence by Prokhorov's theorem (Theorem 5.1.3 [AGS05]), $\mathcal{P}(X)$ is (sequentially) compact. It is then an elementary result that any l.s.c. function attains its minimum on a compact set. □

We next prove Prop. 2.1 regarding the convexity of $\mathcal{E}_\epsilon$ and the uniqueness of its minimum.

Proof of Prop. 2.1. Let μ be any finite signed Borel measure. The compactness assumption and the fact that $p \in \mathcal{C}^1(X)$ imply $p(x) > \delta$ for any $x \in X$ for some $\delta > 0$. Hence $p(x)^{-1/2} \leq \delta^{-1/2}$, so the weighted measure $\tilde{\mu}$ defined by $\mathrm{d}\tilde{\mu}(x) \triangleq p^{-1/2}(x)\mathrm{d}\mu(x)$ is also finite. By the definition of i.s.p.d. kernels, we have $\mathcal{E}_{W_\epsilon}(\mu) = \mathcal{E}_{k_\epsilon}(\tilde{\mu}) > 0$ if $\tilde{\mu}$ is not the zero measure, which is equivalent to μ not being zero since $p > 0$. Thus W_ϵ is i.s.p.d. on X. Also note that $(p(x)p(y))^{-1/2} < \delta^{-1}$ for all $x, y \in X$, so $\mathcal{E}_{W_\epsilon}$ is always finite. By Lemma 1.1 of Pronzato and Zhigljavsky [PZ21], we conclude that $\mathcal{E}_{W_\epsilon}$ is strictly convex in $\mathcal{M}_{\text{sign}}$, the space of finite signed measures, and in particular it is convex on $\mathcal{P}(X)$. Hence combined with the existence result from Lem. 2.5 we conclude $\mathcal{E}_\epsilon$ attains a unique minimum in $\mathcal{P}(X)$. □

Definition 2.6 (Fourier transform)**.** *For $f \in \mathcal{L}^1(\mathbb{R}^n)$, its* Fourier transform $\hat{f}$ *is the complex-valued function defined via*

$$\hat{f}(\xi) \triangleq \int e^{-2\pi i \xi \cdot x} f(x)\mathrm{d}x.$$

More generally, for a signed finite measure $\mu \in \mathcal{M}_{\text{sign}}(\mathbb{R}^n)$, its Fourier transform $\hat{\mu}$ *is the complex-valued function defined via*

$$\hat{\mu}(\xi) \triangleq \int e^{-2\pi i \xi \cdot x} \mathrm{d}\mu(x).$$

This integral is always well-defined and moreover $\hat{\mu}$ is uniformly continuous; see Borodachov, Hardin, and Saff [BHS19, Section 1.10].

We will prove the following weak version (under the additional assumption that a mollifier ϕ is integrable) of Bochner's theorem suitable for our case. In particular we will need the Fourier inversion formula which is not given in the usual statement of Bochner's theorem. On the other hand, we cannot directly use the Fourier inversion formula since it is not obvious how to check the integrability of $\hat{\phi}$ when ϕ is a mollifier.

Lemma 2.6. *Suppose $\phi \in \mathcal{L}^1(\mathbb{R}^n)$ is even, bounded, continuous, and $\boldsymbol{k}(x, y) \triangleq \phi(x - y)$ is i.s.p.d. on any compact sets. Then its Fourier transform $\hat{\phi}$ is real, nonnegative, and the following inversion formula holds:*

$$\phi(x) = \int e^{2\pi i x \cdot \xi} \hat{\phi}(\xi)\mathrm{d}\xi \quad \text{for all } x \in \mathbb{R}^n. \tag{2.15}$$

Proof. The proof is adapted from that of Varadhan [Var01, Theorem 2.7] and is extended to the multi-dimensional case.

Since $\overline{\hat{\phi}(\xi)} \triangleq \int e^{-2\pi i x \cdot \xi} \phi(x)\mathrm{d}x$ and $\phi(x) = \phi(-x)$, with a change of variable $x' = -x$, we obtain $\overline{\hat{\phi}(\xi)} = \hat{\phi}(\xi)$, so $\hat{\phi}$ is real.

Next we show that $\hat{\phi}$ is nonnegative. For $T > 0$, we compute, for a fixed $\xi \in \mathbb{R}^n$,

$$\begin{aligned}
&\frac{1}{T^n}\int_{[0,T]^n}\int_{[0,T]^n} e^{-2\pi i(t-s)\cdot\xi}\phi(t-s)\mathrm{d}t\mathrm{d}s \\
=&\frac{1}{T^n}\int_{[-T,T]^n}\left(\int_{\prod_i[|u_i|,2T-|u_i|]} 2^{-n}\mathrm{d}v\right)e^{-2\pi iu\cdot\xi}\phi(u)\mathrm{d}u \\
=&\frac{1}{T^n}\int_{[-T,T]^n}\left(\prod_{i=1}^n\frac{2T-2\,|u_i|}{2}\right)e^{-2\pi iu\cdot\xi}\phi(u)\mathrm{d}u \\
=&\int_{[-T,T]^n}\left(\prod_{i=1}^n\left(1-\frac{|u_i|}{T}\right)\right)e^{-2\pi iu\cdot\xi}\phi(u)\mathrm{d}u,
\end{aligned}$$

where we have used change of variable $u = t - s$, $v = t + s$. Since $\phi \in \mathcal{L}^1(\mathbb{R}^n)$, by dominated convergence theorem, we have as $T \to \infty$

$$\hat{\phi}(\xi) = \int e^{-2\pi i\xi\cdot x}\phi(x)\mathrm{d}x = \lim_{T\to\infty}\frac{1}{T^n}\int_{[0,T]^n}\int_{[0,T]^n} e^{-2\pi i(t-s)\cdot\xi}\phi(t-s)\mathrm{d}t\mathrm{d}s.$$

For $t, s \in \mathbb{R}^n$, we have

$$\begin{aligned}
&\Re\left(e^{-2\pi i(t-s)\cdot\xi}\phi(t-s))\right) \\
=&\cos(2\pi t\cdot\xi)\boldsymbol{k}(t,s)\cos(2\pi s\cdot\xi) + \sin(2\pi t\cdot\xi)\boldsymbol{k}(t,s)\sin(2\pi s\cdot\xi).
\end{aligned}$$

For a fixed T, if we define a finite measure μ as $\mathrm{d}\mu = \mathbf{1}_{t\in[0,T]^n}\cos(2\pi t\cdot\xi)\mathrm{d}t$, then $\boldsymbol{k}$ being i.s.p.d. implies

$$\iint \phi(t-s)\mathrm{d}\mu\mathrm{d}\mu = \int_{[0,T]^n}\int_{[0,T]^n}\cos(2\pi t\cdot\xi)\phi(t-s)\cos(2\pi s\cdot\xi)\mathrm{d}t\mathrm{d}s \geq 0,$$

and similarly for sin. Since $\hat{\phi}$ is real, we conclude that $\hat{\phi}$ is nonnegative.

Finally we prove (2.15). For $\sigma > 0$, define $\hat{\phi}_\sigma(\xi) \triangleq \hat{\phi}(\xi)e^{-\sigma^2\|\xi\|_2^2}$. Since $\hat{\phi}$ is bounded (because $\phi \in \mathcal{L}^1(\mathbb{R}^n)$), we see that $\hat{\phi}_\sigma \in \mathcal{L}^1(\mathbb{R}^n)$. We compute, for $x \in \mathbb{R}^n$, using Fubini's theorem,

$$\begin{aligned}
\int e^{2\pi ix\cdot\xi}\hat{\phi}_\sigma(\xi)\mathrm{d}\xi &= \int \hat{\phi}(\xi)e^{-\sigma^2\|\xi\|_2^2}e^{2\pi ix\cdot\xi}\mathrm{d}\xi \\
&= \int\left(\int e^{-2\pi iy\cdot\xi}\phi(y)\mathrm{d}y\right)e^{-\sigma^2\|\xi\|_2^2}e^{2\pi ix\cdot\xi}\mathrm{d}\xi \\
&= \int\left(\int e^{-2\pi i(y-x)\cdot\xi}e^{-\sigma^2\|\xi\|_2^2}\mathrm{d}\xi\right)\phi(y)\mathrm{d}y \\
&= \int(\pi/\sigma^2)^{n/2}e^{-\pi^2\|x-y\|_2^2/\sigma^2}\phi(y)\mathrm{d}y,
\end{aligned}$$

where we use the Fourier transform formula [BHS19, (4.4.1)] of the Gaussian distribution. Notice that $p_\sigma(y) \triangleq (\pi/\sigma^2)^{n/2} e^{-\pi^2\|x-y\|_2^2/\sigma^2}$ is the density of a multivariate Gaussian centered at x with covariance $\sigma^2/(2\pi^2)\cdot \boldsymbol{I}$. Hence $\int e^{2\pi i x\cdot\xi}\hat{\phi}_\sigma(\xi)\mathrm{d}\xi = (p_\sigma * \phi)(0)$. Since ϕ is bounded by assumption, with $x = 0$ we find $\int \hat{\phi}_\sigma(\xi)\mathrm{d}\xi \le \|\phi\|_\infty$. Taking $\sigma \to 0$, by monotone convergence theorem, we have $\int \hat{\phi}(\xi)\mathrm{d}\xi \le \|\phi\|_\infty$, so together with the fact that $\hat{\phi} \ge 0$ we have $\hat{\phi} \in \mathcal{L}^1(\mathbb{R}^n)$. Finally, since $\hat{\phi}_\sigma \le \hat{\phi}$, by dominated convergence theorem and Prop. 2.7 (note p_σ is centered at x), we have

$$\int e^{2\pi i x\cdot\xi}\hat{\phi}(\xi)\mathrm{d}\xi = \lim_{\sigma\to 0}\int e^{2\pi i x\cdot\xi}\hat{\phi}_\sigma(\xi)\mathrm{d}\xi = \lim_{\sigma\to 0}(p_\sigma * \phi)(0) = \phi(x).$$

□

Proof of Prop. 2.2. By Lem. 2.6,

$$\begin{aligned}\iint \phi(x-y)\mathrm{d}\nu(x)\mathrm{d}\nu(y) &= \iint \left(\int e^{-2\pi i(x-y)\cdot\xi}\hat{\phi}(\xi)\mathrm{d}\xi\right)\mathrm{d}\nu(x)\mathrm{d}\nu(y)\\ &= \int\left(\int e^{-2\pi i x\cdot\xi}\mathrm{d}\nu(x)\right)\left(\int e^{2\pi i y\cdot\xi}\mathrm{d}\nu(y)\right)\hat{\phi}(\xi)\mathrm{d}\xi\\ &= \int \hat{\nu}(\xi)\overline{\hat{\nu}(\xi)}\hat{\phi}(\xi)\mathrm{d}\xi = \int |\hat{\nu}(\xi)|^2\,\hat{\phi}(\xi)\mathrm{d}\xi.\end{aligned}$$

where we use Fubini's theorem (all measures are finite and $e^{-2\pi i\cdot}$ is bounded) to exchange the order of integration. □

♢ 2.A.3 Differential calculus of $\mathcal{E}_\epsilon$ in $\mathcal{P}_2(\mathbb{R}^n)$

The following lemma for interchanging integration and derivatives will be useful.

Lemma 2.7. *Let $x_0 \in \mathbb{R}$, $h > 0$, and $\Omega \subset \mathbb{R}^m$ be a compact set. Suppose $f : (x_0 - h, x_0 + h)\times\Omega \to \mathbb{R}$ is jointly continuous and the derivative $\frac{\partial}{\partial x}f : (x_0-h, x_0+h)\times\Omega\to\mathbb{R}$ exists and is jointly continuous. Then $\int_\Omega f(x,\omega)\mathrm{d}\omega$ is differentiable for $x \in (x_0-h, x_0+h)$, and*

$$\frac{\mathrm{d}}{\mathrm{d}x}\int_\Omega f(x,\omega)\mathrm{d}\omega = \int_\Omega \frac{\partial}{\partial x}f(x,\omega)\mathrm{d}\omega$$

where the integration is with respect to the Lebesgue measure λ_m on Ω.

Proof. Fix $x \in (x_0 - h, x_0 + h)$ and let $t > 0$ be small enough such that $[x - t, x + t] \subset (x_0 - h, x_0 + h)$. Note that $\int_\Omega f(x,\omega)\mathrm{d}\omega$ is well-defined by the dominated convergence theorem since $\sup_{x\in[x-t,x+t],\omega\in\Omega} f(x,\omega) < \infty$ and $\lambda_m(\Omega) < \infty$. Define $\theta(\omega) \triangleq \sup_{x\in[x-t,x+t]}\left|\frac{\partial}{\partial x}f(x,\omega)\right|$ which is finite since $\frac{\partial}{\partial x}f(x,\omega)$ is continuous, so that $\theta(\omega) \ge \frac{\partial}{\partial x}|f(x,\omega)|$ for all $x \in [x-t, x+t]$ and θ is integrable since $\lambda_m(\Omega) < \infty$. Hence by the differentiation lemma [Kle13, Theorem 6.28] we are done. □

The following lemma is similar to Korba et al. [Kor+21a, Proposition 3] but with different assumptions: we do not put any integrability assumptions on W_ϵ, but we do restrict measures to have compact support. For a differentiable $f : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$, we use $\nabla_1 f(x, y)$ to denote the gradient with respect to x. We also use $\mathrm{H}_1 f(x, y)$ to denote the Hessian of f with respect to x. We similarly denote $\nabla_2 f(x, y)$ and $\mathrm{H}_2 f(x, y)$.

Lemma 2.8. *Assume $\mu \in \mathcal{P}_2(\mathbb{R}^n)$ has density $q \in C^1_c(\mathbb{R}^n)$. Let $\boldsymbol{\xi} \in C(\mathbb{R}^n, \mathbb{R}^n)$. Denote $\boldsymbol{s}(x, t) \triangleq x + t\boldsymbol{\xi}(x)$. Then for all $t > 0$ and all $\epsilon > 0$,*

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathcal{E}_\epsilon(\mu_t) = 2 \iint \left(\boldsymbol{\xi}(x)^\top \nabla_1 W_\epsilon(\boldsymbol{s}(x, t), \boldsymbol{s}(y, t))\right) \mathrm{d}\mu(x)\mathrm{d}\mu(y). \tag{2.16}$$

In particularly,

$$\frac{\mathrm{d}}{\mathrm{d}t}\bigg|_{t=0} \mathcal{E}_\epsilon(\mu_t) = 2 \iint \left(\boldsymbol{\xi}(x)^\top \nabla_1 W_\epsilon(x, y)\right) \mathrm{d}\mu(x)\mathrm{d}\mu(y).$$

Moreover,

$$\frac{\mathrm{d}^2}{\mathrm{d}t^2}\bigg|_{t=0} \mathcal{E}_\epsilon(\mu_t) = 2 \iint \left(\boldsymbol{\xi}(x)^\top \nabla_1 \nabla_2 W_\epsilon(x, y)\boldsymbol{\xi}(y) + \boldsymbol{\xi}(x)^\top \mathrm{H}_1 W_\epsilon(x, y)\boldsymbol{\xi}(x)\right) \mathrm{d}\mu(x)\mathrm{d}\mu(y). \tag{2.17}$$

Proof. We compute

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathcal{E}_\epsilon(\mu_t) = \frac{\mathrm{d}}{\mathrm{d}t}\left(2 \iint W_\epsilon(\boldsymbol{s}(x, t), \boldsymbol{s}(y, t))\mathrm{d}\mu(x)\mathrm{d}\mu(y)\right).$$

Since μ has compact support and $(x, y, t) \mapsto W_\epsilon(\boldsymbol{s}(x, t), \boldsymbol{s}(y, t))q(x)q(y)$ is jointly continuous and its derivative with respect to t is also jointly continuous, by Lem. 2.7, we can push $\frac{\mathrm{d}}{\mathrm{d}t}$ inside the double integral and we obtain (2.16). Another application of Lem. 2.7 shows that if we take derivative with respect to t again on (2.16) and evaluate at 0 we obtain (2.17). □

Lemma 2.9. *Let $f \in \mathcal{C}^1(\mathbb{R}^n)$, $p \in [1, \infty]$. Assume ϕ_ϵ has compact support for some $\epsilon > 0$. Then for all $i = 1, \ldots, n$, $\phi_\epsilon * f$ is differentiable and*

$$\frac{\partial}{\partial x_i}(\phi_\epsilon * f)(x) = \left(\phi_\epsilon * \frac{\partial}{\partial x_i} f\right)(x).$$

Proof. Since $\mathrm{supp}(\phi_\epsilon)$ is compact and $f \in \mathcal{C}^1(\mathbb{R}^n)$ is bounded on any compact set, we know $\phi_\epsilon * f$ is well-defined at every $x \in \mathbb{R}^n$. Note that

$$\begin{aligned}\frac{\partial}{\partial x_i}(\phi_\epsilon * f)(x) &= \frac{\partial}{\partial x_i}\left(\int f(x - y)\phi_\epsilon(y)\mathrm{d}y\right) \\ &\overset{(?)}{=} \int \frac{\partial}{\partial f} x_i(x - y)\phi_\epsilon(y)\mathrm{d}y = \left(\phi_\epsilon * \frac{\partial}{\partial x_i} f\right)(x).\end{aligned}$$

Since $\mathrm{supp}(\phi_\epsilon)$ is compact and $(x,y) \mapsto f(x-y)\phi(y)$ is $\mathcal{C}^1$, by Lem. 2.7 we justify the existence of the derivative and the exchange of differentiation and integration (?). □

Subdifferentials of $\mathcal{E}_\epsilon$

Recall the following notion of a "Wasserstein gradient" in $\mathcal{P}_2(\mathbb{R}^n)$ from Ambrosio, Gigli, and Savaré [AGS05, Definition 10.1.1].

Definition 2.7. *A vector field $\boldsymbol{w} \in \mathcal{L}^2(\mu, \mathbb{R}^n)$ is a* strong Fréchet subdifferential *of a functional $\mathcal{F} : \mathcal{P}_2(\mathbb{R}^n) \to (-\infty, +\infty]$ if for all $\boldsymbol{T} \in \mathcal{L}^2(\mu, \mathbb{R}^n)$, the following holds:*

$$\mathcal{F}(\boldsymbol{T}_\# \mu) - \mathcal{F}(\mu) \geq \int \boldsymbol{w}(x)^\top (\boldsymbol{T}(x) - x) \mathrm{d}\mu(x) + o\left(\|\boldsymbol{T} - \boldsymbol{I}\|_{\mathcal{L}^2(\mu, \mathbb{R}^n)}\right). \tag{2.18}$$

Note that we cannot apply Ambrosio, Gigli, and Savaré [AGS05, Lemma 10.4.1] directly to prove (2.5) because interaction energies cannot be written in the form of (10.4.1) in their setup.

Proof of Prop. 2.3. Let $\boldsymbol{\xi} \in \mathcal{C}_c^\infty(\mathbb{R}^n, \mathbb{R}^n)$. By Lem. 2.8, we have

$$\begin{aligned}
\left.\frac{\mathrm{d}}{\mathrm{d}t}\right|_{t=0} \mathcal{E}_\epsilon(\mu_t) &= 2 \iint \left(\boldsymbol{\xi}(x)^\top \nabla_1 W_\epsilon(x,y)\right) \mathrm{d}\mu(x) \mathrm{d}\mu(y) \\
&= 2 \iint \left(\boldsymbol{\xi}(x)^\top \nabla_1 \left(\phi_\epsilon(x-y)(p(x)p(y))^{-1/2}\right)\right) q(y) \mathrm{d}y \mathrm{d}\mu(x) \\
&= 2 \int \boldsymbol{\xi}(x)^\top \nabla \left(p(x)^{-1/2} (\phi_\epsilon * q/\sqrt{p})(x)\right) \mathrm{d}\mu(x),
\end{aligned}$$

where the last step follows from applying Lem. 2.7 since q has compact support. Now suppose $\boldsymbol{w} \in \mathcal{L}^2(\mu, \mathbb{R}^n)$ is a strong Fréchet subdifferential satisfying (2.18). For the sequence $\{\boldsymbol{T}_t\}$, we have by definition

$$\begin{aligned}
\mathcal{E}_\epsilon(\mu_t) - \mathcal{E}_\epsilon(\mu) &\geq \int \boldsymbol{w}(x)^\top (\boldsymbol{T}_t(x) - x) \mathrm{d}\mu(x) + o\left(\|\boldsymbol{T}_t - \boldsymbol{I}\|_{\mathcal{L}^2(\mu, \mathbb{R}^n)}\right) \\
&= \int \boldsymbol{w}(x)^\top (t\boldsymbol{\xi}(x)) \mathrm{d}\mu(x) + o(t).
\end{aligned}$$

Hence

$$\liminf_{t \downarrow 0} \frac{\mathcal{E}_\epsilon(\mu_t) - \mathcal{E}_\epsilon(\mu)}{t} \geq \int \boldsymbol{w}(x)^\top \boldsymbol{\xi}(x) \mathrm{d}\mu(x) \geq \liminf_{t \uparrow 0} \frac{\mathcal{E}_\epsilon(\mu_t) - \mathcal{E}_\epsilon(\mu)}{t}.$$

The previous calculation shows $\frac{\mathrm{d}}{\mathrm{d}t}|_{t=0} \mathcal{E}_\epsilon(\mu_t)$ exists, and hence it is equal to $\int \boldsymbol{w}(x)^\top \boldsymbol{\xi}(x) \mathrm{d}\mu(x)$. This proves for any $\boldsymbol{\xi} \in C_c^\infty(\mathbb{R}^n, \mathbb{R}^n)$,

$$\int \boldsymbol{w}(x)^\top \boldsymbol{\xi}(x) \mathrm{d}\mu(x) = \int \boldsymbol{\xi}(x)^\top \nabla \left(2p(x)^{-1/2} (\phi_\epsilon * q/\sqrt{p})(x)\right) \mathrm{d}\mu(x).$$

Hence we have shown (2.5).

Finally, we show $\lim_{\epsilon\to 0} \boldsymbol{w}_\epsilon(x) = \boldsymbol{w}_{\chi^2}(x)$ for μ-a.e. x under the additional assumption that ϕ_ϵ has compact support. By Ambrosio, Gigli, and Savaré [AGS05, Lemma 10.4.1] with $F(x,\rho(x)) = \left(\frac{\rho(x)}{p(x)} - 1\right)^2 p(x)$, we find the strong subdifferential of $\chi^2(\cdot \parallel \mu^*)$ is given by

$$\boldsymbol{w}_{\chi^2,\mu}(x) = 2\nabla \frac{q(x)}{p(x)}, \text{ for } \mu\text{-a.e. } x \in \mathbb{R}^n.$$

To show (2.6), we compute, for μ-a.e. x,

$$\begin{aligned}
\boldsymbol{w}_\epsilon(x) &= \nabla \left(p(x)^{-1/2} (\phi_\epsilon * q/\sqrt{p})(x) \right) \\
&= \nabla (p(x)^{-1/2})(\phi_\epsilon * q/\sqrt{p})(x) + p(x)^{-1/2} \nabla(\phi_\epsilon * q/\sqrt{p})(x) \\
&= \nabla (p(x)^{-1/2})(\phi_\epsilon * q/\sqrt{p})(x) + p(x)^{-1/2} (\phi_\epsilon * \nabla(q/\sqrt{p}))(x),
\end{aligned}$$

where for the last equality we have applied Lem. 2.9. Now taking $\epsilon \to 0$, by Prop. 2.7 using the fact that $\operatorname{supp}(\phi_\epsilon)$ is compact (so that $q/\sqrt{p}$ and $\nabla(q/\sqrt{p})$ are bounded on $x + \operatorname{supp}(\phi_\epsilon)$), we obtain (2.6). □

Displacement convexity of $\mathcal{E}_\epsilon$ at μ^* as $\epsilon \to 0$

The statement of Prop. 2.4 is similar in form as Korba et al. [Kor+21a, Corollary 4] but in our case we do not have the second term in (2.17) vanishing and we need to take the limit $\epsilon \to 0$. We also do not resort to RKHS theory in the proof.

Proof of Prop. 2.4. By (2.17), we have

$$\left.\frac{\mathrm{d}^2}{\mathrm{d}t^2}\right|_{t=0} \mathcal{E}_\epsilon(\mu_t) = 2(\mathcal{F}_\epsilon + \mathcal{G}_\epsilon),$$

where

$$\begin{aligned}
F_\epsilon &= \iint \left(\boldsymbol{\xi}(x)^\top \nabla_1 \nabla_2 W_\epsilon(x,y) \boldsymbol{\xi}(y) \right) \mathrm{d}\mu^*(x) \mathrm{d}\mu^*(y) \\
G_\epsilon &= \iint \left(\boldsymbol{\xi}(x)^\top \mathrm{H}_1 W_\epsilon(x,y) \boldsymbol{\xi}(x) \right) \mathrm{d}\mu^*(x) \mathrm{d}\mu^*(y).
\end{aligned}$$

We tackle F_ϵ first. Observe that successive application of integration by parts using the

fact that $\boldsymbol{\xi}$ has compact support gives

$$\begin{aligned}
F_\epsilon &= \sum_{i,j=1}^n \iint \boldsymbol{\xi}_i(x) \frac{\partial}{\partial x_i} y_j W_\epsilon(x,y) \boldsymbol{\xi}_j(y) p(x) p(y) \mathrm{d}x\mathrm{d}y \\
&= -\sum_{i,j=1}^n \iint \frac{\partial}{\partial x_i}(\boldsymbol{\xi}_i(x)p(x)) \frac{\partial}{\partial y_j} W_\epsilon(x,y) \boldsymbol{\xi}_j(y) p(y) \mathrm{d}x\mathrm{d}y \\
&= \sum_{i,j=1}^n \iint \frac{\partial}{\partial x_i}(\boldsymbol{\xi}_i(x)p(x)) W_\epsilon(x,y) \frac{\partial}{\partial y_j}(\boldsymbol{\xi}_j(y)p(y)) \mathrm{d}x\mathrm{d}y \\
&= \iint \left(\sum_{i=1}^n \frac{\partial}{\partial x_i}(\boldsymbol{\xi}_i(x)p(x)) \right) W_\epsilon(x,y) \left(\sum_{j=1}^n \frac{\partial}{\partial y_j}(\boldsymbol{\xi}_j(y)p(y)) \right) \mathrm{d}x\mathrm{d}y.
\end{aligned}$$

If we view $\sum_{i=1}^n \frac{\partial}{\partial x_i}(\boldsymbol{\xi}_i(x)p(x))$ as the density of a signed measure (it is integrable since it has compact support), and since W_ϵ is i.s.p.d. on the support of $\boldsymbol{\xi}$ by Prop. 2.1(a), we see that each double integral in the last expression is non-negative. Hence $F_\epsilon \geq 0$.

Next we show $\lim_{\epsilon\to 0} G_\epsilon = 0$. Since μ^* has compact support, by Fubini's theorem,

$$G_\epsilon = \int \boldsymbol{\xi}(x)^\top \mathrm{H}_1 \left(\int W_\epsilon(x,y) p(y) \mathrm{d}y \right) \boldsymbol{\xi}(x) p(x) \mathrm{d}x.$$

To expand the integral inside the Hessian operator, we have

$$\mathrm{H}_1 \left(\int W_\epsilon(x,y) p(y) \mathrm{d}y \right) = \mathrm{H} \left(p(x)^{-1/2} (\phi_\epsilon * \sqrt{p})(x) \right).$$

First by the chain rule and Lem. 2.9, we have

$$\frac{\mathrm{d}}{\mathrm{d}x} \left(p(x)^{-1/2} (\phi_\epsilon * \sqrt{p})(x) \right) = \frac{\mathrm{d}}{\mathrm{d}x} \left(p(x)^{-1/2} \right) (\phi_\epsilon * \sqrt{p}) + p(x)^{-1/2} \left(\phi_\epsilon * \frac{\mathrm{d}}{\mathrm{d}x} \sqrt{p}(x) \right).$$

Differentiating again while applying Lem. 2.9, we obtain after rearranging terms,

$$\begin{aligned}
\mathrm{H}_1 \left(\int W_\epsilon(x,y) p(y) \mathrm{d}y \right) = &\left(\frac{\mathrm{d}^2}{\mathrm{d}x^2} p(x)^{-1/2} \right) (\phi_\epsilon * \sqrt{p})(x) \\
&+ 2 \left(\frac{\mathrm{d}}{\mathrm{d}x} p(x)^{-1/2} \right) \left(\phi_\epsilon * \frac{\mathrm{d}}{\mathrm{d}x} \sqrt{p} \right)(x) \\
&+ p(x)^{-1/2} \left(\phi_\epsilon * \frac{\mathrm{d}^2}{\mathrm{d}x^2} \sqrt{p} \right)(x). \qquad (2.19)
\end{aligned}$$

By Prop. 2.7 and the fact that $p \in \mathcal{C}_c^2(\mathbb{R}^n)$, we have

$$\begin{aligned}
&\lim_{\epsilon\to 0} \mathrm{H}_1 \left(\int W_\epsilon(x,y)p(y)\mathrm{d}y \right) \\
=&\left(\frac{\mathrm{d}^2}{\mathrm{d}x^2} p(x)^{-1/2} \right) \sqrt{p}(x) + 2 \left(\frac{\mathrm{d}}{\mathrm{d}x} p(x)^{-1/2} \right) \frac{\mathrm{d}}{\mathrm{d}x} \sqrt{p}(x) + p(x)^{-1/2} \frac{\mathrm{d}^2}{\mathrm{d}x^2} \sqrt{p}(x) \\
=&\frac{\mathrm{d}^2}{\mathrm{d}x^2} \left(p(x)^{-1/2} \sqrt{p(x)} \right) = 0.
\end{aligned}$$

Finally, we have

$$\begin{aligned}
\lim_{\epsilon\to 0} G_\epsilon &= \lim_{\epsilon\to 0} \int \boldsymbol{\xi}(x)^\top \mathrm{H}_1 \left(\int W_\epsilon(x,y)p(y)\mathrm{d}y \right) \boldsymbol{\xi}(x)p(x)\mathrm{d}x \\
&= \int \boldsymbol{\xi}(x)^\top \left(\lim_{\epsilon\to 0} \mathrm{H}_1 \left(\int W_\epsilon(x,y)p(y)\mathrm{d}y \right) \right) \boldsymbol{\xi}(x)p(x)\mathrm{d}x \\
&= 0,
\end{aligned}$$

where interchanging the limit and the integral is jusfitied by the dominated convergence theorem and the fact that p and ϕ_ϵ have compact support (we need compact support assumption of ϕ_ϵ to make sure convolutions appearing in (2.19) are uniformly bounded when ϵ is sufficiently small).

□

A descent lemma for $\mathcal{E}_\epsilon$ with time discretization

Lemma 2.10. *Under Assum. 2.1, for $L > 0$ defined in (2.8), the function $\nabla_1 W_\epsilon : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^n$ is L-Lipschitz in terms of $\|\cdot\|_2$ in either input.*

Proof. Denote $r(x) \triangleq p(x)^{-1/2}$. Then our assumptions imply $r(x) \le C_r \triangleq C_p^{1/2}$, $\|\nabla r(x)\|_2 \le C_r' \triangleq \frac{1}{2} C_p^{3/2} C_p'$, and $\|\mathrm{H}_1\, r(x)\|_2 \le C_r'' \triangleq \frac{3}{4} C_p^{5/2} C_p'^2 + \frac{1}{2} C_p^{3/2} C_p''$. We compute, for $x, y \in \mathbb{R}^n$,

$$\begin{aligned}
\nabla_1 W_\epsilon(x,y) &= \nabla_x \left(\phi_\epsilon(x-y) r(x) r(y) \right) \\
&= \nabla \phi_\epsilon(x-y) r(x) r(y) + \phi_\epsilon(x-y) \nabla r(x) r(y).
\end{aligned}$$

Then

$$\mathrm{H}_1\, W_\epsilon(x,y) = \mathrm{H}\, \phi_\epsilon(x-y) r(x) r(y) + 2 \nabla \phi_\epsilon(x-y) \nabla r(x)^\top r(y) + \phi_\epsilon(x-y)\, \mathrm{H}\, r(x) r(y),$$

and

$$\begin{aligned}
\nabla_2 \nabla_1 W_\epsilon(x,y) = &- \mathrm{H}\, \phi_\epsilon(x-y) r(x) r(y) + \nabla \phi_\epsilon(x-y) r(x) \nabla r(y)^\top \\
&- \nabla \phi_\epsilon(x-y) \nabla r(x) r(y) + \phi_\epsilon(x-y) \nabla r(x) \nabla r(y)^\top.
\end{aligned}$$

Then we have

$$\|\mathrm{H}_1 W_\epsilon(x,y)\|_2 \le C_\epsilon'' C_r^2 + 2C_\epsilon' C_r' C_r + C_\epsilon C_r'' C_r,$$
$$\|\nabla_2 \nabla_1 W_\epsilon(x,y)\|_2 \le C_\epsilon'' C_r^2 + 2C_\epsilon' C_r C_r' + C_\epsilon C_r'^2.$$

Hence we conclude that $\nabla_1 W_\epsilon$ is L-Lipschitz with L defined in (2.8).

□

Remark 2.3. *Compared with Korba et al. [Kor+21a, Lemma 1], to ensure $\nabla_1 W_\epsilon$ is Lipscthiz, we only require uniform boundedness of p and ϕ_ϵ up to second order derivatives instead of up to order 3.*

Proof of Prop. 2.5. We first show μ_m has compact support for all $m \in \mathbb{N}_0$. For $\mu \in \mathcal{P}_2(\mathbb{R}^n)$ with compact support, the proof of Prop. 2.3 implies (note Prop. 2.3 assumes μ has density but it is not necessary to obtain the following formula using the same proof)

$$\boldsymbol{w}_{\epsilon,\mu}(x) = 2\int \nabla_1 W_\epsilon(x,y)\mathrm{d}\mu(y).$$

Since $x \mapsto \nabla_1 W_\epsilon(x,y)$ is $\mathcal{C}^1$ by the assumptions and μ has compact support, by Lem. 2.7, we see that $\boldsymbol{w}_{\epsilon,\mu}$ is differentiable with

$$\nabla \boldsymbol{w}_{\epsilon,\mu}(x) = 2\int \mathrm{H}_1 W_\epsilon(x,y)\mathrm{d}\mu(y).$$

By induction, suppose μ_m has compact support. Then

$$\mathrm{supp}(\mu_{m+1}) \subset (\boldsymbol{I} - \gamma \boldsymbol{w}_{\epsilon,\mu_m})_{\#}\,\mathrm{supp}(\mu_m) \subset \mathrm{supp}(\mu_m) + \gamma\left(\sup_{x\in\mathrm{supp}(\mu_m)} \|\boldsymbol{w}_{\epsilon,\mu_m}\|_2\right)\mathcal{B}_2(1).$$

Since $\boldsymbol{w}_{\epsilon,\mu_m}$ is continuous, the set on the right-hand side is bounded. Hence μ_{m+1} has compact support.

Now fix $m \in \mathbb{N}_0$ and we show (2.9). Define a path $\{\mu_t\}_{t\in[0,1]}$ defined by $\mu_t = (\boldsymbol{I} - \gamma t \boldsymbol{w}_{\epsilon,\mu_m})_{\#}\mu_m$. Let $f(t) \triangleq \mathcal{E}_\epsilon(\mu_t)$. By Lem. 2.8, the continuity of $\boldsymbol{w}_{\epsilon,\mu_m}$ implies that f is differentiable. Moreover, another application of Lem. 2.7 implies that f is twice differentiable, and in particular continuously differentiable. Hence f is absolutely continuous on the compact interval $[0,1]$. By the fundamental theorem of calculus, we have

$$\mathcal{E}_\epsilon(\mu_{m+1}) - \mathcal{E}_\epsilon(\mu_m) = f(1) - f(0) = f'(0) + \int_0^1 (f'(t) - f'(0))\mathrm{d}t.$$

Observe that by Lem. 2.8,

$$\begin{aligned} f'(0) &= 2\iint (-\gamma \boldsymbol{w}_{\epsilon,\mu_m}(x))^\top \nabla_1 W_\epsilon(x,y)\mathrm{d}\mu_m(x)\mathrm{d}\mu_m(y) \\ &= \int (-\gamma \boldsymbol{w}_{\epsilon,\mu_m}(x))\left(2\int \nabla_1 W_\epsilon(x,y)\mathrm{d}\mu_m(y)\right)\mathrm{d}\mu_m(x) \\ &= -\gamma \|\boldsymbol{w}_{\epsilon,\mu_m}(x)\|^2_{\mathcal{L}^2(\mu_m)}, \end{aligned}$$

where we apply Fubini's theorem to exchange the order of integration together with the fact that μ_m has compact support and the integrand is continuous. Let $\boldsymbol{s}(x,t) \triangleq x - \gamma t \boldsymbol{w}_{\epsilon,\mu_m}(x)$. Note that, again by Lem. 2.8,

$$\begin{aligned}|f'(t) - f'(0)| &\le 2 \iint \left|(-\gamma \boldsymbol{w}_{\epsilon,\mu_m}(x))^\top \left(\nabla_1 W_\epsilon(\boldsymbol{s}(x,t), \boldsymbol{s}(y,t)) - \nabla_1 W_\epsilon(x,y)\right)\right| \mathrm{d}\mu_m(x)\mathrm{d}\mu_m(y) \\ &\le 2\gamma \iint \|\boldsymbol{w}_{\epsilon,\mu_m}(x)\|_2 \|\nabla_1 W_\epsilon(\boldsymbol{s}(x,t), \boldsymbol{s}(y,t)) - \nabla_1 W_\epsilon(x,y)\|_2 \mathrm{d}\mu_m(x)\mathrm{d}\mu_m(y).\end{aligned}$$

Note that, by Lem. 2.10, we have

$$\begin{aligned}&\|\nabla_1 W_\epsilon(\boldsymbol{s}(x,t), \boldsymbol{s}(y,t)) - \nabla_1 W_\epsilon(x,y)\|_2 \\ \le&\|\nabla_1 W_\epsilon(\boldsymbol{s}(x,t), \boldsymbol{s}(y,t)) - \nabla_1 W_\epsilon(\boldsymbol{s}(x,t),y)\|_2 + \|\nabla_1 W_\epsilon(\boldsymbol{s}(x,t),y) - \nabla_1 W_\epsilon(x,y)\|_2 \\ \le& L\gamma t(\|\boldsymbol{w}_{\epsilon,\mu_m}(x)\|_2 + \|\boldsymbol{w}_{\epsilon,\mu_m}(y)\|_2).\end{aligned}$$

Thus

$$\begin{aligned}|f'(t) - f'(0)| &\le 2\gamma^2 Lt \iint \|\boldsymbol{w}_{\epsilon,\mu_m}(x)\|_2 \left(\|\boldsymbol{w}_{\epsilon,\mu_m}(x)\|_2 + \|\boldsymbol{w}_{\epsilon,\mu_m}(y)\|_2\right) \mathrm{d}\mu_m(x)\mathrm{d}\mu_m(y) \\ &\le 2\gamma^2 Lt \left(\|\boldsymbol{w}_{\epsilon,\mu_m}\|_{\mathcal{L}^2(\mu_m)} + \left(\int \|\boldsymbol{w}_{\epsilon,\mu_m}(x)\|_2 \mathrm{d}\mu_m(x)\right)^2\right) \\ &\le 4\gamma^2 Lt \|\boldsymbol{w}_{\epsilon,\mu_m}\|_{\mathcal{L}^2(\mu_m)},\end{aligned}$$

where we have used the Cauchy-Schwartz inequality in the last step.

Combining everything, we have shown

$$\begin{aligned}\mathcal{E}_\epsilon(\mu_{m+1}) - \mathcal{E}_\epsilon(\mu_m) &= f'(0) + \int_0^1 (f'(t) - f'(0))\mathrm{d}t \\ &\le -\gamma(1 - 2\gamma L)\|\boldsymbol{w}_{\epsilon,\mu_m}\|_{\mathcal{L}^2(\mu_m)} \le 0,\end{aligned}$$

since $\gamma < \frac{1}{2L}$.

□

■ 2.B Weighted Hypersingular Riesz Energy

We recall results most relevant to us from Borodachov, Hardin, and Saff [BHS19]. Suppose $X \subset \mathbb{R}^n$ is compact, of Hausdorff dimension d, and d-rectifiable, i.e., the image of a bounded set in $\mathbb{R}^d$ under a Lipschitz mapping. Given a non-vanishing continuous probability density p on X, define a measure $\mathcal{H}_d^p(B) \triangleq \int_B p(x)\mathrm{d}\mathcal{H}_d(x)$ for any Borel set $B \subset X$. The target measure h_d^p is defined to be $h_d^p(B) \triangleq \frac{\mathcal{H}_d^p(B)}{\mathcal{H}_d^p(X)}$. The *weighted s-Riesz energy* is defined as the interaction energy

$$E_s(\omega_N) \triangleq \sum_{i \ne j} \frac{(p(x_i)p(x_j))^{-s/2d}}{\|x_i - x_j\|^s}. \tag{2.20}$$

The following result states that minimizers of the weighted s-Riesz energy approximate the target measure when the s is sufficiently large.

Theorem 2.3 (Theorem 11.1.2, Borodachov, Hardin, and Saff [BHS19]). *Suppose $s > d$. For $\omega_N^* \in \arg\min E_s$ with $\omega_N^* = \{x_1^N, \dots, x_N^N\}$, we have weak convergence $\delta_{\omega_N^*} \to h_d^p$ as $N \to \infty$.*

A similar result with a slightly different assumption holds for $s = d$ (Theorem 11.1.3 of Borodachov, Hardin, and Saff [BHS19]).

Comparison of (2.20) with discrete MIE (2.10). Our theory is only valid for the full-dimensional case, i.e., $n = d$ (see a few exceptions discussed in Rem. 2.2). When this is the case and if the mollifier is taken to be from the s-Riesz family, (2.10) becomes, for $s > n$,

$$E_\epsilon(\omega_N) \triangleq \sum_{i,j=1}^N \frac{(p(x_i)p(x_j))^{-1/2}}{(\|x_i - x_j\|_2^2 + \epsilon^2)^{s/2}}.$$

Compared with (2.20), in our case there is an ϵ in the denominator, so that the continuous version of the energy $\mathcal{E}_\epsilon$ does not blow up all the time (this is in contrast with the continuous version of (2.20)—see Borodachov, Hardin, and Saff [BHS19, Theorem 4.3.1]). Moreover, the exponential scaling on p is different: in our case we use $-1/2$ whereas in (2.20) it is $-s/2n$. The diagonal $i = j$ is included in our energy, but this is inconsequential as another valid discretization is to discard the diagonal term [BHS19, Theorem 4.2.2]. On the other hand, our theory allows a bigger class of mollifiers that are not necessarily of Riesz families. We also allow X to be non-compact. An interesting future research direction is to extend our theory to cases where $d < n$, e.g., when X is an embedded d-dimensional submanifold in $\mathbb{R}^n$.

■ 2.C Algorithmic Details

In this section we provide algorithmic details of MIED and compare with the updates of SVGD [LW16]. The negative gradient of (2.11) with respect to x_i is

$$\begin{aligned}
-\nabla_{x_i} \log E_\epsilon(\omega_N) =& 2\sum_{j\neq i} \frac{e^{I_{ij}}}{\sum_{i,j} e^{I_{ij}}} \left((\nabla \log \phi_\epsilon)(x_j - x_i) + \frac{1}{2}\nabla \log p(x_i) \right) \\
&+ \frac{e^{I_{ii}}}{\sum_{i,j} e^{I_{ij}}} \nabla \log p(x_i) \\
=& \sum_{j=1}^N \frac{e^{I_{ij}}}{\sum_{i,j} e^{I_{ij}}} \left(2\nabla \log \phi_\epsilon(x_j - x_i) + \nabla \log p(x_i) \right),
\end{aligned}$$

where

$$I_{ij} \triangleq \log \phi_\epsilon(x_i - x_j) - \frac{1}{2}(\log p(x_i) + \log p(x_j)),$$

and to get the last equality we used the fact that $\nabla\phi(0) = 0$ thanks to the assumption $\phi(x) = \phi(-x)$. Then gradient descent on (2.11) gives our algorithm (Alg. 2.1). The special treatment of the diagonal terms described in Sec. 2.4 amounts to modifying only the diagonal I_{ii}.

♢ 2.C.1 Comparison with SVGD

The update formula in Alg. 2.1 is similar to the one in SVGD: if we use $\phi_\epsilon(x-y)$ in place of the kernel $k(x, y)$ in the SVGD update and rewrite:

$$\begin{aligned} x_i^{t+1} &= x_i^t + \eta \sum_{j=1}^N (\nabla\phi_\epsilon(x_j - x_i) + \phi_\epsilon(x_j - x_i)\nabla \log p(x_j)) \\ &= x_i^t + \eta \sum_{j=1}^N \phi_\epsilon(x_j - x_i) \left(\nabla \log \phi_\epsilon(x_j - x_i) + \nabla \log p(x_j)\right). \end{aligned} \tag{SVGD}$$

For both algorithms, the update formula for each particle consists of attraction and repulsion terms and the total time complexity of each update iteration is $O(N^2)$. We note the following differences. First, in our formulation we have scaling factors $\frac{e^{I_{ij}}}{\sum_{i,j} e^{I_{ij}}}$ which help stabilizing the optimization (as a by-product of working in the logarithmic domain) and put more weight on nearby particles as well as particles in low-density regions, whereas in (SVGD) the scaling factors are $\phi_\epsilon(x_j - x_i)$ which are not adapted to prioritize low-density regions. Second, in MIED, the attraction force for particle i only comes from $\nabla \log p(x_i)$, whereas in (SVGD) the attraction force comes from $\nabla \log p(x_j)$ for all j's. Third, for each j, in our formulation the repulsive force has an additional factor of 2 in front of $\nabla \log \phi_\epsilon(x_j - x_i)$.

Empirically, the additional scaling factors in MIED help produce samples with good separations compared to SVGD, since closer pairs of points will have large weights. Additionally, since MIED optimizes a finite-dimensional objective (2.11), we can employ accelerated gradient-based optimizers like Adam [KB14], which we used in our experiments. In contrast, SVGD does not optimize any finite-dimensional objective. While practical SVGD implementations also use optimizers like Adam, it is unclear how the resulting particle dynamics is related to the gradient flow of KL divergence.

♢ 2.C.2 Handling constraints with dynamic barrier method

The dynamic barrier method is introduced in Gong and Liu [GL21] which solves $\min_x f(x)$ subject to $g(x) \le 0$ where g is scalar-valued. Intuitively, their method computes update

directions by either decreasing $g(x)$ when $g(x) > 0$, following $-\nabla f(x)$ if the constraints are satisfied, or balancing both gradient directions.

In order to handle multiple constraints such as in Fig. 2.D.7, we consider a generalized version of their dynamic barrier method. In this generalized setting, $g : \mathbb{R}^n \to \mathbb{R}^m$ is vector-valued and constraints are $g(x) \leq 0$ where the $\leq$ sign is interpreted coordinate wise.. Suppose we are at iteration t with current solution x^t. Then the next update direction v^* is taken to be the arg min of

$$\min_{v \in \mathbb{R}^n} \|v - \nabla f(x^t)\|_2^2 \text{ s.t. } \forall i = 1, \ldots, m, \nabla g_i(x^t)^\top v \geq \alpha_i g_i(x^t), \tag{2.21}$$

where $\alpha_i > 0$ are fixed hyperparameters; in our implementation we simply choose $\alpha_i = 1$. Then $x^{t+1} = x^t - \eta v^*$ with learning rate η. In our implementation we use Adam [KB14] that modulates the update directions. Observe that the optimization problem (2.21) is the same as projecting a point $\nabla f(x^t)$ onto the polyhedron formed by the intersection of the halfspaces $\cap_{i=1}^m \{x \in \mathbb{R}^n : \nabla g_i(x^t)^\top v \geq \alpha_i g_i(x^t)\}$. To solve (2.21), we use Dykstra's algorithm [Tib17] which can be interpreted as running coordinate descent on the dual of (2.21). We use a fixed number of 20 iterations for the Dykstra's algorithm which we found to be sufficient for our experiments; in the case of a single constraint, we only need to use one iteration.

■ 2.D Experiment Details and Additional Results

♢ 2.D.1 Gaussians in varying dimensions

We generate the n-dimensional Gaussians used to produce Fig. 2.1 as follows. We generate a matrix $\sqrt{A} \in \mathbb{R}^{n \times n}$ with i.i.d. entries uniformly in $[-1, 1]$. Then we set $A = \sqrt{A}\sqrt{A}^\top / \det(\sqrt{A})$. This way $\det(A) = 1$. We then use A as the covariance matrix for the Gaussian (centered at 0). We use Adam with learning rate 0.01 for all methods for a total of 2000 iterations. This is enough for SVGD and MIED to converge, while for KSDD the convergence can be much slower.

We visualize the samples from each method for $n = 2$ in Fig. 2.D.1. We notice that MIED is capable of generating well-separated samples, while for SVGD there is a gap between the inner cluster of samples and a sparse outer ring. For KSDD we see the artifact where too many samples concentrate on the diagonal.

♢ 2.D.2 Collapsed samples when the kernel width is too big

In Fig. 2.4, we see that samples from SVGD collapse with an adaptive kernel where the variance is taken to be half of the median of the squared distance among all pairs of points [LW16]; at termination the median of the squared distance is greater than 1 in that

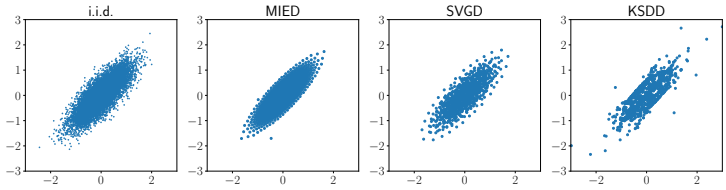

Figure 2.D.1: Visualization of samples from each method for a 2D skewed Gaussian.

experiment. Here we investigate this issue further. In Fig. 2.D.2, for the same uniform sampling setup, we visualize the results of SVGD with a fixed kernel width and MIED with Gaussian mollifiers with the same kernel width: when the kernel width (i.e. $2\epsilon^2$ in $\phi_\epsilon^g(x) = \exp(-\frac{\|x\|_2^2}{2\epsilon^2})/Z_\epsilon^g$) is too big, both SVGD and MIED result in collapsed samples. This is because since the target density is a constant, the only force in the updates is the repulsive force. When the kernel width is too large, the repulsive force coming from points in the same collapsed cluster is dominated by the repulsive force coming from points from other clusters—this is evident in the update directions shown in the leftmost column of Fig. 2.D.2 (SVGD with kernel width 0.1). When using the dynamic barrier

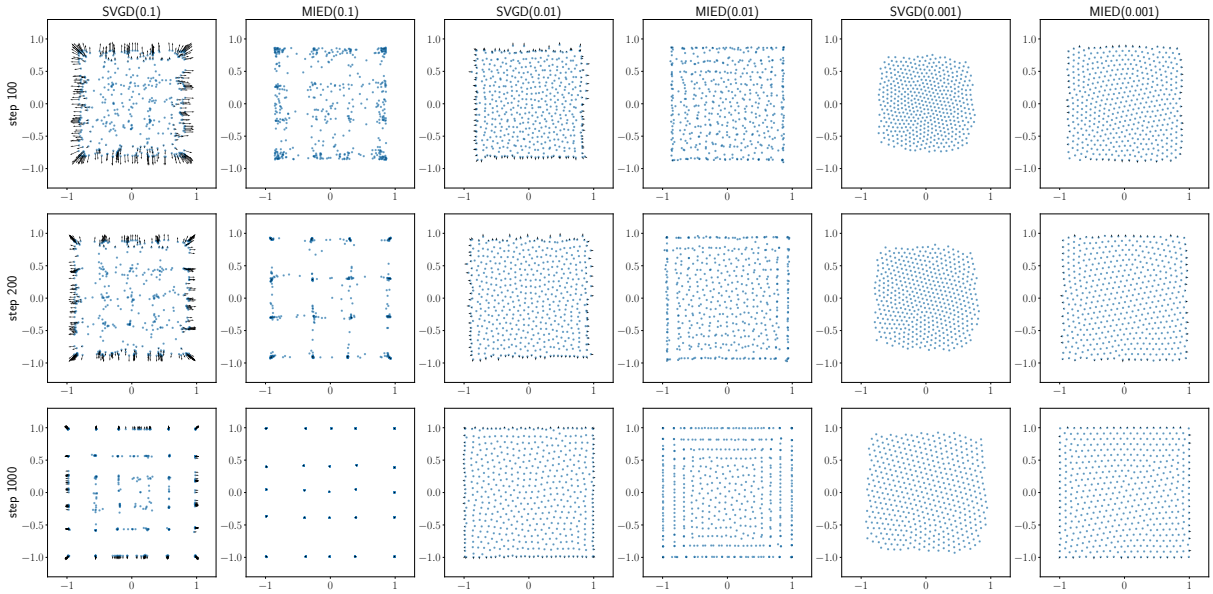

Figure 2.D.2: SVGD and MIED with fixed-size Gaussian kernels for uniform sampling in a square. In each cell of the grid, we plot the samples along with the update direction (black arrows) at that iteration. Rows correspond to iterations $100, 200, 1000$. Columns correspond to each method with varying kernel widths (twice the variance of the Gaussian kernel/mollifier) indicated in the parentheses.

method [GL21] to enforce the square constraints instead of reparameterization with tanh, we obtain similar results as in Fig. 2.D.2.

This pathological phenomenon is not only limited to constrained sampling: when sampling the 2D Gaussian from Fig. 2.D.1, using too big a kernel width can also result in collapsing (Fig. 2.D.3).

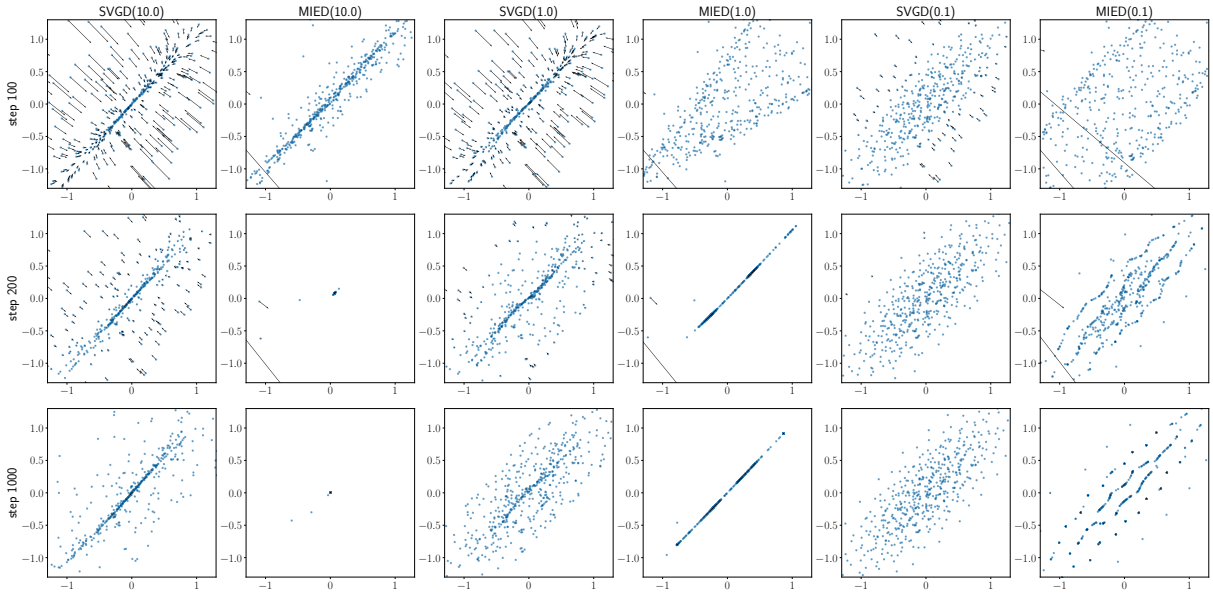

Figure 2.D.3: SVGD and MIED with fixed-size Gaussian kernels for sampling a 2D Gaussian as in Fig. 2.D.1. We see that using too big a kernel size can lead to collapsed samples for both SVGD and MIED.

We emphasize that our theory of MIED suggests that in practice we need to choose the kernel width very small in order to sample from the correct target measure according to Thm. 2.1 and Thm. 2.2. In comparison, the theory of SVGD has no such implication.

♢ 2.D.3 Uniform sampling with an alternative mirror map

In this section, we show that for sampling from a uniform distribution in the square $[-1,1]^2$, the results of SVMD/MSVGD [SLM21] depend heavily on the choice of the mirror map. Instead of the entropic mirror map used to produce results in Figs. 2.3 and 2.4, here we use the mirror map $\phi(\theta) = \sum_{i=1}^n \left(\log \frac{1}{1-\theta_i} + \log \frac{1}{1+\theta_i}\right)$ as in Ahn and Chewi [AC21]. The results are shown in Fig. 2.D.4. SVMD/MSVGD fail to draw samples near the boundary; we suspect this is because the gradient of the conjugate $\nabla\phi^*(\eta) = \frac{(\sqrt{1+\eta^2}-1)}{\eta}$ (coordinate-wise arithmetic) requires coordinates of η to go to ∞ to land near the boundary. We verify this phenomenon by using $\nabla\phi^*(\eta)$ as the reparametrization map in MIED (rightmost

figure in Fig. 2.D.4): indeed with such reparameterization MIED also struggles near the boundary.

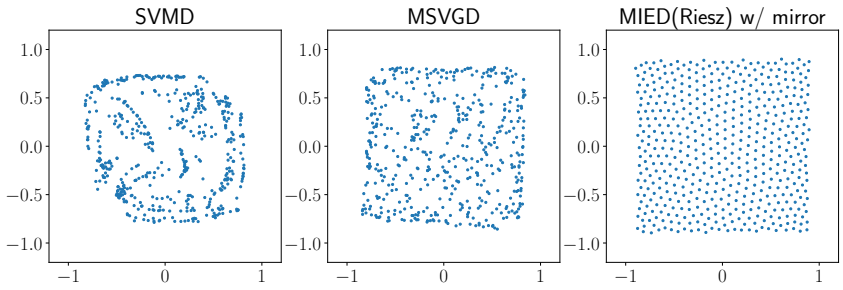

Figure 2.D.4: Visualization of samples for uniform sampling from a 2D box when using a suboptimal mirror map. All three methods fail to draw samples near the boundary of the box $[-1, 1]^2$.

♢ 2.D.4 20-dimensional Dirichlet distribution

We sample from the 20-dimensional Dirichlet distribution in the same setup as in Shi, Liu, and Mackey [SLM21] with 50 particles. Results and visualization are shown in Fig. 2.D.5. We see that unlike sampling from a box (Fig. 2.3), both MSVGD and SVMD by Shi, Liu, and Mackey [SLM21] perform better than MIED. This is due to the fact that the entropic mirror map used here is a well-tested choice for simplex constraint, yet obtaining a good mirror map for a generic constraint, even if it is linear, can be challenging. Our method does not have such a limitation, as it can easily incorporate existing constrained optimization tools.

♢ 2.D.5 Effect of s for Riesz mollifiers

When we use the s-Riesz families of mollifiers in MIED, we have the freedom of choosing the hyperparameter s so long as $s > n$. In this section, we study the effect of s on the resulting samples. We consider the problem of sampling from a mixture of four 2D Gaussians centered at $(\pm 1, \pm 1)$, each with diagonal variance 0.3 and constrained to the $[-1, 1]^2$ box. We vary s in $[2, 10]$ and the number of particles N in $\{100, 200, 500, 1000, 2000\}$. All runs use a total of 1000 iterations with learning rate 0.01. In the top of Fig. 2.D.6, we plot the W_2 distance and energy distance as functions of s for each N. Interestingly, we see the best performing s is in $[3, 5.0]$ and depends on N. This suggests that our choice of $s = n + 10^{-4}$ in Sec. 2.5 may not be optimal and there is room for hyperparameter tuning to further improve the performance of MIED with Riesz kernel. At the bottom of Fig. 2.D.6 we visualize the samples of MIED with $s = 3$ and of SVGD with kernel width

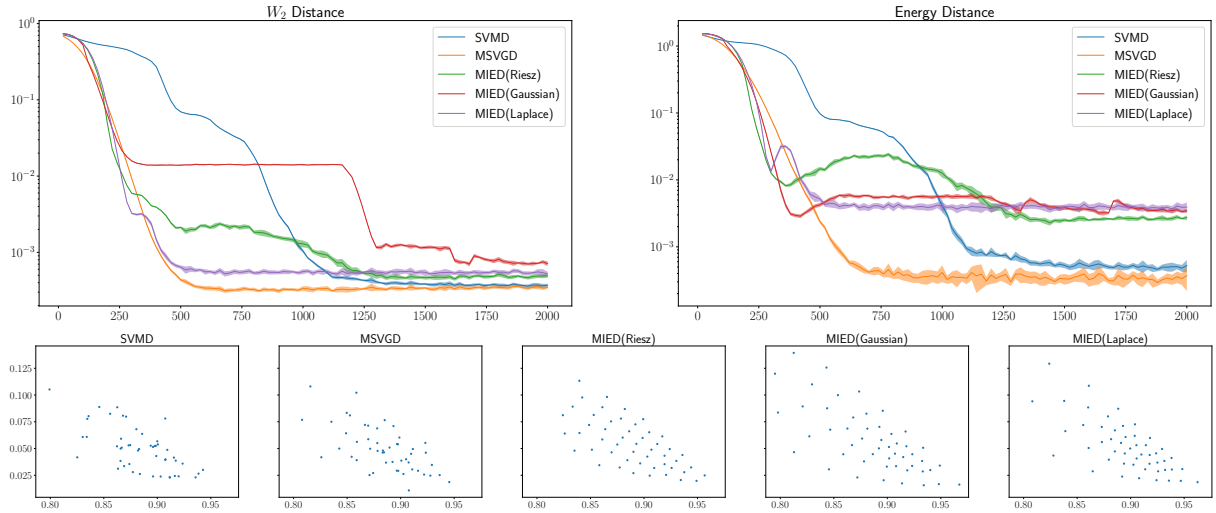

Figure 2.D.5: Top: Metrics vs. the number of iteration for sampling the 20-dimensional Dirichlet distribution. Bottom: visualization of samples from each method.

0.01 (adaptive kernels would result in collapsed samples and other widths we tested on would result in worse samples). While SVGD samples form visible artifacts, the samples of MIED are evenly distributed and the four modes of the mixtures emerge as N increases.

♢ 2.D.6 More constrained sampling experiments

In this section we test MIED on more low dimensional constrained sampling problems and qualitatively assess the results. Note that mirror LMC [Zha+20; AC21] or mirror SVGD [SLM21] cannot be applied due to non-convexity of the constraints. In Fig. 2.D.7, we consider uniform sampling of a challenging 2D region with initial samples drawn from the top-right corner: as the number of iterations increases, MIED gradually propagate samples to fill up the entire region. In Fig. 2.D.8, we consider sampling from a von Mises-Fisher distribution on a unit sphere. Although our theory focuses on sampling from a full-dimensional distribution, as discussed in Rem. 2.2, we can extend Thm. 2.1 to the case of a sphere due to its symmetry. We see the samples visualized in Fig. 2.D.8 capture the two modes that emerge by restricting the Gaussian to a unit sphere.

♢ 2.D.7 Details on fairness Bayesian neural network experiment

We use 80%/20% training/test split as in Liu, Tong, and Liu [LTL21]. We use the source code provided by Liu, Tong, and Liu [LTL21] with default hyperparameters.The source code provided by the authors of Liu, Tong, and Liu [LTL21] does not implement the calculation of $g(\theta)$ faithfully as written in the formula, so we corrected it. All methods

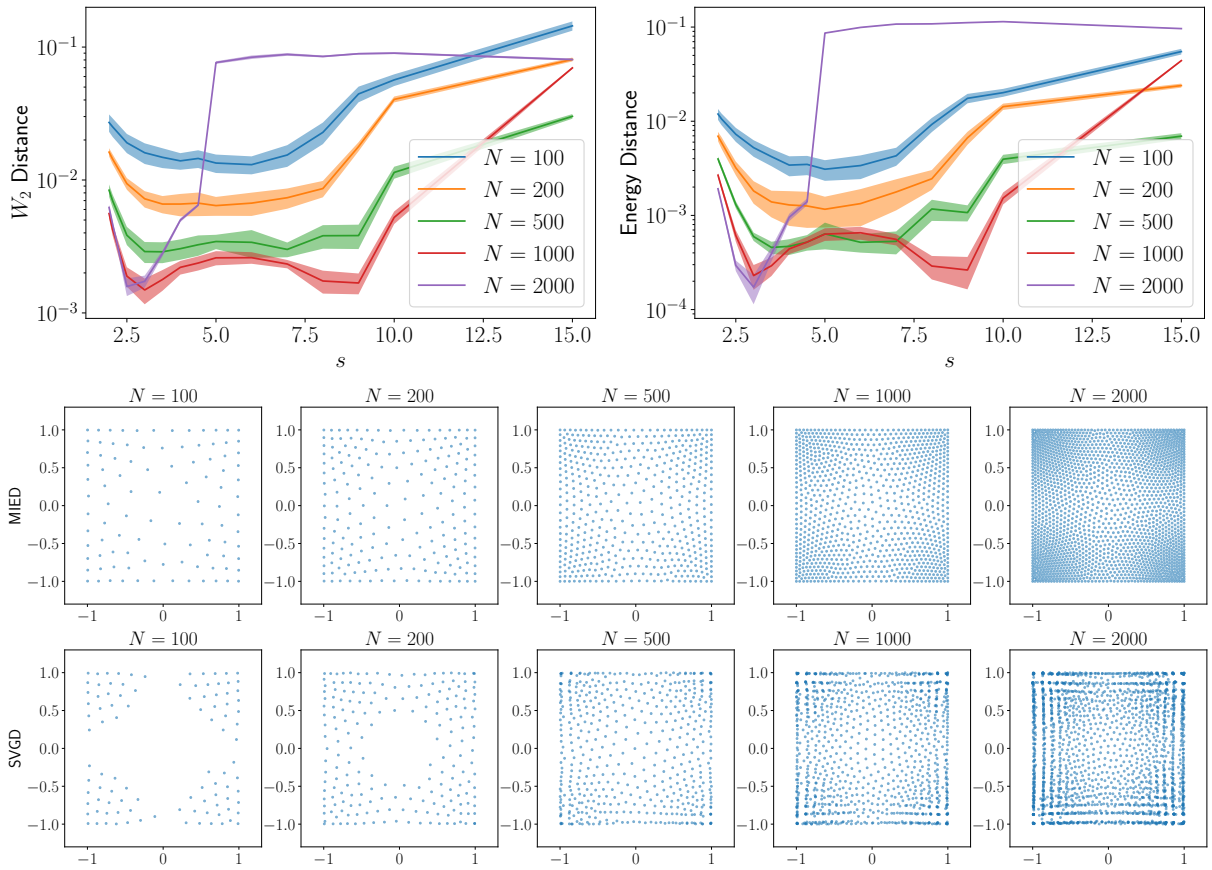

Figure 2.D.6: Ablation study of the hyperparameter s for MIED with s-Riesz families of mollifiers. Top: metrics as functions of s. Bottom: visualization of samples of MIED with a Riesz mollifier with $s = 3$ and of SVGD with kernel width 0.01.

use 2000 iterations for training. For our method we use learning rate 0.001. One of their four methods (Control+SVGD) got stuck at initialization (with accuracy around 0.75), so we omit its result from the plot in Fig. 2.5.

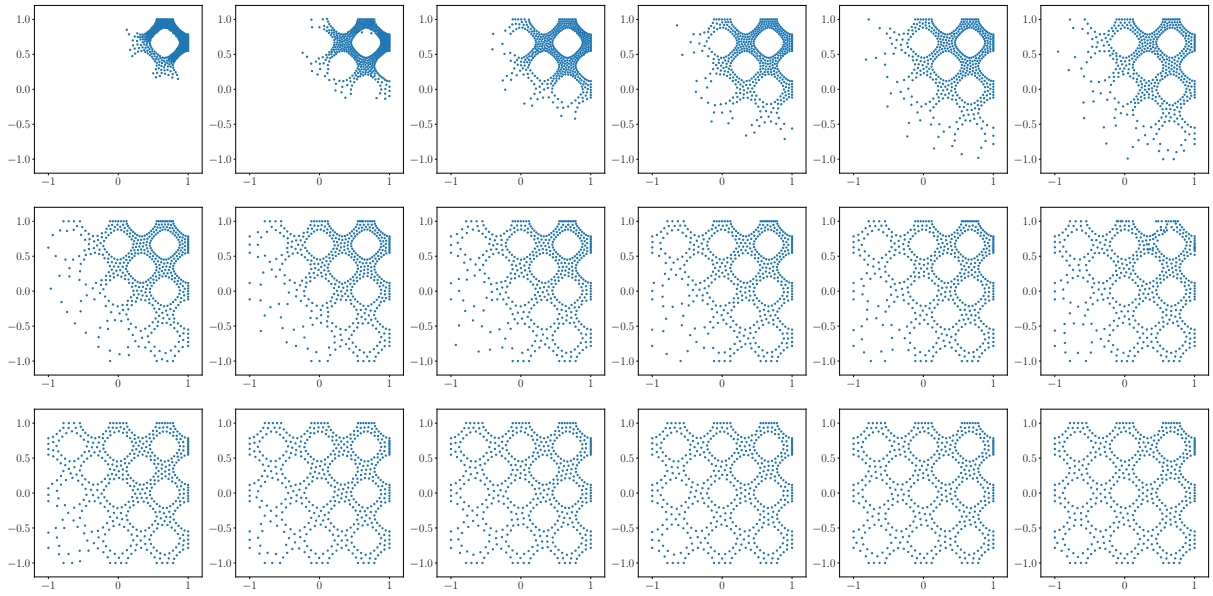

Figure 2.D.7: Uniform sampling of the region $\{(x, y) \in [-1, 1]^2 : (\cos(3\pi x) + \cos(3\pi y))^2 < 0.3\}$ using MIED with a Riesz mollifier ($s = 3$) where the constraint is enforced using the dynamic barrier method. The plot in row i column j shows the samples at iteration $100 + 200(6i + j)$. The initial samples are drawn uniformly from the top-right square $[0.5, 1.0]^2$.

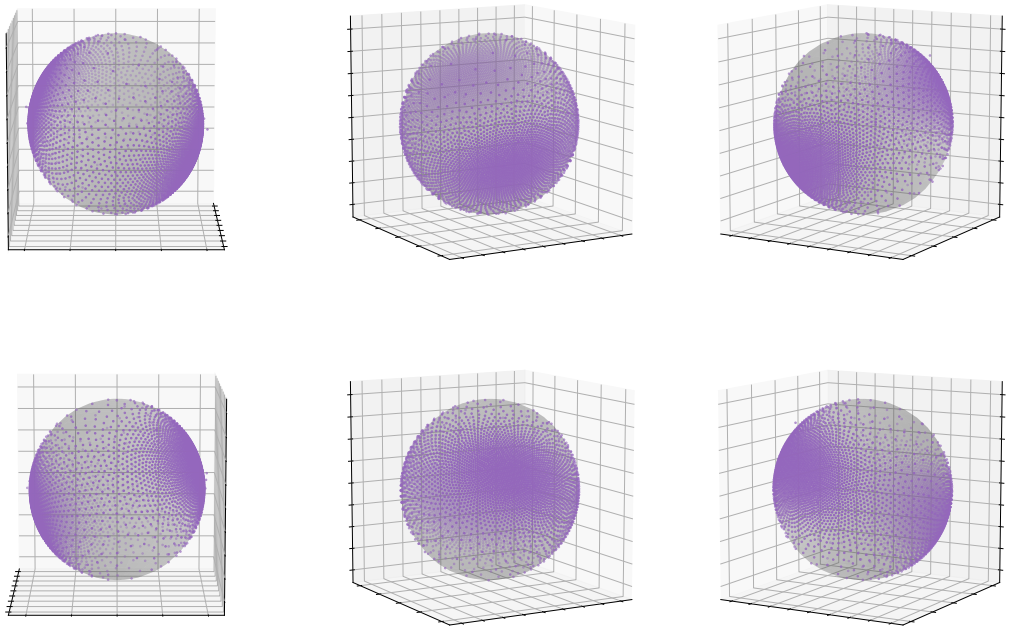

Figure 2.D.8: Sampling from the von Mises-Fisher distribution obtained by constraining the 3-dimensional Gaussian from Sec. 2.D.1 to the unit sphere. The unit-sphere constraint is enforced using the dynamic barrier method and the shown results are obtained using MIED with Riesz kernel and $s = 3$. The six plots are views from six evenly spaced azimuthal angles.

Chapter 3

Debiased Distribution Compression

In Chapter 2, we explored how minimizing an interactive energy of the form $\sum_{i,j=1}^{n} \boldsymbol{k}(x_i, x_j)$ over particles provides a good approximation of the target distribution $\mathbb{P}$ by establishing a rich connection with the χ^2 divergence. However, this approach has its limitations: each gradient step takes $\Theta(n^2)$ time to compute, and moreover, the finite-dimensional optimization problem over moving particles is inherently non-convex.

In this chapter, we shift our perspective from optimizing over moving particles to optimizing an m-sparse weight w for fixed particles given as input:

$$\min_{w:\|w\|_0 \leq m} \sum_{i,j=1}^{n} w_i w_j \boldsymbol{k}(x_i, x_j). \tag{3.1}$$

In other words, we will look for a *coreset* of size $m \ll n$ that simultaneously debiases and compresses the input points $(x_i)_{i=1}^n$. Despite the non-convex sparsity constraint in (3.1), we demonstrate that this problem can be solved efficiently, yielding a heavily compressed coreset with accurarcy comparable to n i.i.d. unbiased samples within quadratic or even sub-quadratic time in n. This chapter is based on the publication [LDM24].

■ 3.1 Introduction

Distribution compression is the problem of summarizing a target probability distribution $\mathbb{P}$ with a small set of representative points. Such compact summaries are particularly valuable for tasks that incur substantial downstream computation costs per summary point, like organ and tissue modeling in which each simulation consumes thousands of CPU hours [Nie+11].

Remarkably, modern compression methods can summarize a distribution more succinctly than i.i.d. sampling. For example, kernel thinning (KT) [DM21; DM22b], Compress++ [SDM22], recombination [HOL23], and randomly pivoted Cholesky [EM24] all

Table 1: **Methods for debiased distribution compression.** For each method, we report the smallest coreset size m and running time, up to logarithmic factors, sufficient to guarantee $\widetilde{O}(n^{-1/2})$ $\mathrm{MMD}_{\boldsymbol{k}_{\mathbb{P}}}$ to $\mathbb{P}$ given a LogGrowth kernel $\boldsymbol{k}_{\mathbb{P}}$ and n slow-growing input points $\mathcal{S}_n = (x_i)_{i=1}^n$ from a fast-mixing Markov chain targeting $\mathbb{Q}$ with tails no lighter than $\mathbb{P}$ (see Thm. 3.1 and Def. 3.3). For generic slow-growing $\mathcal{S}_n$, identical guarantees hold for excess $\mathrm{MMD}_{\boldsymbol{k}_{\mathbb{P}}}$ (3.4) relative to the best simplex reweighting of $\mathcal{S}_n$.

Method		Compression Type	Coreset Size m	Runtime	Source
Stein Thinning [Ria+22]		equal-weighted	n	$d_{\boldsymbol{k}_{\mathbb{P}}} n^2$	Sec. 3.D.1
Stein Kernel Thinning	Greedy (Alg. 3.2)	equal-weighted	$\sqrt{n}$	$d_{\boldsymbol{k}_{\mathbb{P}}} n^2$	Thm. 3.3
	Low-rank (Alg. 3.4)			$d_{\boldsymbol{k}_{\mathbb{P}}} n^{1.5}$	Thm. 3.5
Stein Recombination	Greedy	simplex-weighted	poly-log(n)	$d_{\boldsymbol{k}_{\mathbb{P}}} n^2$	Thm. 3.6
	Low-rank (Alg. 3.6)			$d_{\boldsymbol{k}_{\mathbb{P}}} n + n^{1.5}$	
Stein Cholesky	Greedy	constant-preserving	poly-log(n)	$d_{\boldsymbol{k}_{\mathbb{P}}} n^2$	Thm. 3.7
	Low-rank (Alg. 3.8)			$d_{\boldsymbol{k}_{\mathbb{P}}} n + n^{1.5}$	

provide $\widetilde{O}(1/m)$ approximation error using m points, a significant improvement over the $\Omega(1/\sqrt{m})$ approximation provided by i.i.d. sampling from $\mathbb{P}$. However, each of these constructions relies on access to an accurate input sequence, like an i.i.d. sample from $\mathbb{P}$ or a Markov chain converging quickly to $\mathbb{P}$.

Much more commonly, one only has access to n *biased* sample points approximating a wrong distribution $\mathbb{Q}$. Such biases are a common occurrence in Markov chain Monte Carlo (MCMC)-based inference due to tempering [where one targets a less peaked and more dispersed distribution to achieve faster convergence, GSK10], burn-in [where the initial state of a Markov chain biases the distribution of chain iterates, CC96], or approximate MCMC [where one runs a cheaper approximate Markov chain to avoid the prohibitive costs of an exact MCMC algorithm, e.g., AKW12]. The Stein thinning (ST) method of Riabiz et al. [Ria+22] was developed to provide accurate compression even when the input sample sequence provides a poor approximation to the target. ST operates by greedily thinning the input sample to minimize the maximum mean discrepancy [MMD, Gre+12] to $\mathbb{P}$. However, ST is only known to provide an $O(1/\sqrt{m})$ approximation to $\mathbb{P}$; this guarantee is no better than that of i.i.d. sampling and a far cry from the $\widetilde{O}(1/m)$ error achieved with unbiased coreset constructions.

In this work, we address this deficit by developing new, efficient coreset constructions that provably yield better-than-i.i.d. error even when the input sample is biased. For $\mathbb{P}$ on $\mathbb{R}^d$, our primary contributions are fourfold and summarized in Tab. 1. First, for the task of equal-weighted compression, we introduce *Stein Kernel Thinning* (SKT, Alg. 3.2), a strategy that combines the greedy bias correction properties of ST with the unbiased compression of KT to produce $\sqrt{n}$ summary points with error $\widetilde{O}(n^{-1/2})$ in $O(n^2)$ time. In

contrast, ST would require $\Omega(n)$ points to guarantee this error. Second, for larger-scale compression problems, we propose *Low-rank SKT* (Alg. 3.4), a strategy that combines the scalable summarization of Compress++ with a new low-rank debiasing procedure (Alg. 3.3) to match the SKT guarantees in sub-quadratic $o(n^2)$ time.

Third, for the task of simplex-weighted compression, in which summary points are accompanied by weights in the simplex, we propose greedy and low-rank *Stein Recombination* (Alg. 3.6) constructions that match the guarantees of SKT with as few as poly-$\log(n)$ points. Finally, for the task of constant-preserving compression, in which summary points are accompanied by real-valued weights summing to 1, we introduce greedy and low-rank *Stein Cholesky* (Alg. 3.8) constructions that again match the guarantees of SKT using as few as poly-$\log(n)$ points.

Underlying these advances are new guarantees for the quality of simplex-weighted coresets (Thms. 3.1 and 3.2), the spectral decay of kernel matrices (Cor. 3.B.1), and the covering numbers of Stein kernel Hilbert spaces (Prop. 3.1) that may be of independent interest. In Sec. 3.5, we employ our new procedures to produce compact summaries of complex target distributions given input points biased by burn-in, approximate MCMC, or tempering.

Notation We assume Borel-measurable sets and functions and define $[n] \triangleq \{1, \dots, n\}$, $\Delta_{n-1} \triangleq \{w \in \mathbb{R}^n : w \geq 0, \mathbf{1}^\top w = 1\}$, $\|x\|_0 \triangleq |\{i : x_i \neq 0\}|$, and $\|x\|_p^p \triangleq \sum_i |x_i|^p$ for $x \in \mathbb{R}^d$ and $p \geq 1$. For $x \in \mathbb{R}^d$, δ_x denotes the delta measure at x. We let $\mathcal{H}_{\boldsymbol{k}}$ denote the reproducing kernel Hilbert space (RKHS) of a kernel $\boldsymbol{k} : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ [Aro50] and $\|f\|_{\boldsymbol{k}}$ denote the RKHS norm of $f \in \mathcal{H}_{\boldsymbol{k}}$. For a measure μ and separately μ-integrable $\boldsymbol{k}$ and f, we write $\mu f \triangleq \int f(x) \mathrm{d}\mu(x)$ and $\mu \boldsymbol{k}(x) \triangleq \int \boldsymbol{k}(x, y) \mathrm{d}\mu(y)$. The divergence of a differentiable matrix-valued function A is $(\nabla_x \cdot A(x))_j = \sum_i \partial_{x_i} A_{ij}(x)$. For random variables $(X_n)_{n \in \mathbb{N}}$, we say $X_n = O(f(n, \delta))$ holds with probability $\geq 1 - \delta$ if $\Pr(X_n \leq C f(n, \delta)) \geq 1 - \delta$ for a constant C independent of (n, δ) and all n sufficiently large. When using this notation, we view all algorithm parameters except δ as functions of n. For $A \in \mathbb{R}^{n \times n}$ and $v \in \mathbb{R}^n$, $\mathrm{diag}(A)$ and $\mathrm{diag}(v)$ are $n \times n$ diagonal matrices with A_{ii} and v_i respectively as the i-th diagonal entry.

■ 3.2 Debiased Distribution Compression

Throughout, we aim to summarize a fixed target distribution $\mathbb{P}$ on $\mathbb{R}^d$ using a sequence $\mathcal{S}_n \triangleq (x_i)_{i=1}^n$ of potentially biased candidate points in $\mathbb{R}^d$.[1] Correcting for unknown biases in $\mathcal{S}_n$ requires some auxiliary knowledge of $\mathbb{P}$. For us, this knowledge comes in the form of a kernel function $\boldsymbol{k}_{\mathbb{P}}$ with known expectation under $\mathbb{P}$. Without loss of generality, we

[1] Our coreset constructions will in fact apply to any sample space, but our analysis will focus on $\mathbb{R}^d$.

can take this kernel mean to be identically zero.[2]

Assumption 3.1 (Mean-zero kernel)**.** *For some* $\mathfrak{p} \geq 1/2$, $\mathbb{E}_{x\sim\mathbb{P}}[\boldsymbol{k}_{\mathbb{P}}(x,x)^{\mathfrak{p}}] < \infty$ *and* $\mathbb{P}\boldsymbol{k}_{\mathbb{P}} \equiv 0$.

Given a target compression size m, our goal is to output an weight vector $w \in \mathbb{R}^n$ with $\|w\|_0 \leq m$, $\mathbf{1}_n^\top w = 1$, and $o(m^{-1/2})$ (better-than-i.i.d.) maximum mean discrepancy (MMD) to $\mathbb{P}$:

$$\mathrm{MMD}_{\boldsymbol{k}_{\mathbb{P}}}(\textstyle\sum_{i=1}^n w_i\delta_{x_i}, \mathbb{P}) \triangleq \sqrt{\sum_{i,j=1}^n w_i w_j \boldsymbol{k}_{\mathbb{P}}(x_i,x_j)}. \tag{3.2}$$

We consider three standard compression tasks with $\|w\|_0 \leq m$. In *equal-weighted compression* one selects m possibly repeated points from $\mathcal{S}_n$ and assigns each a weight of $\frac{1}{m}$; because of repeats, the induced weight vector over $\mathcal{S}_n$ satisfies $w \in \Delta_{n-1} \cap (\frac{\mathbb{N}_0}{m})^n$. In *simplex-weighted compression* we allow any $w \in \Delta_{n-1}$, and in *constant-preserving compression* we simply enforce $\mathbf{1}_n^\top w = 1$.

When making big O statements, we will treat $\mathcal{S}_n$ as the prefix of an infinite sequence $\mathcal{S}_\infty \triangleq (x_i)_{i\in\mathbb{N}}$. We also write $\boldsymbol{k}_{\mathbb{P}}(\mathcal{S}_n[\mathtt{J}], \mathcal{S}_n[\mathtt{J}]) \triangleq [\boldsymbol{k}_{\mathbb{P}}(x_i,x_j)]_{i,j\in\mathtt{J}}$ for the principal kernel submatrix with indices $\mathtt{J} \subseteq [n]$.

♢ 3.2.1 Kernel assumptions

Many practical *Stein kernel* constructions are available for generating mean-zero kernels for a target $\mathbb{P}$ [CSG16; LLJ16; GM17; Gor+19; Bar+19; Yan+18; AM23]. We will use the most prominent of these Stein kernels as a running example:

Definition 3.1 (Stein kernel)**.** *Given a differentiable base kernel* $\boldsymbol{k}$ *and a symmetric positive semidefinite matrix* M*, the* Stein kernel $\boldsymbol{k}_p : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ *for* $\mathbb{P}$ *with positive differentiable Lebesgue density* p *is defined as*

$$\boldsymbol{k}_p(x,y) \triangleq \tfrac{1}{p(x)p(y)}\nabla_x \cdot \nabla_y \cdot (p(x)M\boldsymbol{k}(x,y)p(y)).$$

While our algorithms apply to any mean zero kernel, our guarantees adapt to the underlying smoothness of the kernels. Our next definition and assumption make this precise.

Definition 3.2 (Covering number)**.** *For a kernel* $\boldsymbol{k} : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ *with* $\mathcal{B}_{\boldsymbol{k}} \triangleq \{f \in \mathcal{H}_{\boldsymbol{k}} : \|f\|_{\boldsymbol{k}} \leq 1\}$*, a set* $A \subset \mathbb{R}^d$*, and* $\varepsilon > 0$*, the covering number* $\mathcal{N}_{\boldsymbol{k}}(A,\varepsilon)$ *is the minimum cardinality of all sets* $\mathcal{C} \subset \mathcal{B}_{\boldsymbol{k}}$ *satisfying*

$$\mathcal{B}_{\boldsymbol{k}} \subset \textstyle\bigcup_{h\in\mathcal{C}}\{g \in \mathcal{B}_{\boldsymbol{k}} : \sup_{x\in A}|h(x)-g(x)| \leq \varepsilon\}.$$

[2] For $\mathbb{P}\boldsymbol{k}_{\mathbb{P}} \not\equiv 0$, the kernel $\boldsymbol{k}_{\mathbb{P}}'(x,y) = \boldsymbol{k}_{\mathbb{P}}(x,y) - \mathbb{P}\boldsymbol{k}_{\mathbb{P}}(x) - \mathbb{P}\boldsymbol{k}_{\mathbb{P}}(y) + \mathbb{P}\mathbb{P}\boldsymbol{k}_{\mathbb{P}}$ satisfies $\mathbb{P}\boldsymbol{k}_{\mathbb{P}}' \equiv 0$ and $\mathrm{MMD}_{\boldsymbol{k}_{\mathbb{P}}'} = \mathrm{MMD}_{\boldsymbol{k}_{\mathbb{P}}}$.

Assumption (α, β)-kernel. *For some $\mathfrak{C}_d > 0$, all $r > 0$ and $\varepsilon \in (0, 1)$, and $\mathcal{B}_2(r) \triangleq \{x \in \mathbb{R}^d : \|x\|_2 \leq r\}$, a kernel $\boldsymbol{k}$ is either* PolyGrowth(α, β)*, i.e.,*

$$\log \mathcal{N}_{\boldsymbol{k}}(\mathcal{B}_2(r), \varepsilon) \leq \mathfrak{C}_d (1/\varepsilon)^{\alpha} (r+1)^{\beta},$$

with $\alpha < 2$ or LogGrowth(α, β)*, i.e.,*

$$\log \mathcal{N}_{\boldsymbol{k}}(\mathcal{B}_2(r), \varepsilon) \leq \mathfrak{C}_d \log(e/\varepsilon)^{\alpha} (r+1)^{\beta}.$$

In Cor. 3.B.1 we show that the eigenvalues of kernel matrices with PolyGrowth and LogGrowth kernels have polynomial and exponential decay respectively. Dwivedi and Mackey [DM22b, Prop. 2] showed that all sufficiently differentiable kernels satisfy the PolyGrowth condition and that bounded radially analytic kernels are LogGrowth. Our next result, proved in Sec. 3.B.2, shows that a Stein kernel $\boldsymbol{k}_p$ can inherit the growth properties of its base kernel even if $\boldsymbol{k}_p$ is itself unbounded and non-smooth.

Proposition 3.1 (Stein kernel growth rates)**.** *A Stein kernel $\boldsymbol{k}_p$ with*

$$\sup_{\|x\|_2 \leq r} \|\nabla \log p(x)\|_2 = O(r^{d_\ell}),$$

for $d_\ell \geq 0$, is

(a) LogGrowth$(d+1, 2d+\delta)$ *for any $\delta > 0$ if the base kernel $\boldsymbol{k}$ is radially analytic (Def. 3.B.3) and*

(b) PolyGrowth$(\frac{d}{s-1}, (1+\frac{d_\ell}{s})d)$ *if the base kernel $\boldsymbol{k}$ is s-times continuously differentiable (Def. 3.B.2) for $s > 1$.*

Notably, the popular Gaussian (Ex. 3.B.1) and inverse multiquadric (Ex. 3.B.2) base kernels satisfy the LogGrowth preconditions, while Matérn, B-spline, sinc, sech, and Wendland's compactly supported kernels satisfy the PolyGrowth precondition [DM22b, Prop. 3]. To our knowledge, Prop. 3.1 provides the first covering number bounds and eigenvalue decay rates for the (typically unbounded) Stein kernels $\boldsymbol{k}_p$.

♢ 3.2.2 Input point desiderata

Our primary desideratum for the input points is that they can be debiased into an accurate estimate of $\mathbb{P}$. Indeed, our high-level strategy for debiased compression is to first use $\boldsymbol{k}_{\mathbb{P}}$ to debias the input points into a more accurate approximation of $\mathbb{P}$ and then compress that approximation into a more succinct representation. Fortunately, even when the input $\mathcal{S}_n$ targets a distribution $\mathbb{Q} \neq \mathbb{P}$, effective debiasing is often achievable via simplex reweighting, i.e., by solving the convex optimization problem

$$\begin{aligned} &w_{\text{OPT}} \in \arg\min_{w \in \Delta_{n-1}} \textstyle\sum_{i,j=1}^n w_i w_j \boldsymbol{k}_{\mathbb{P}}(x_i, x_j) \\ &\text{with} \quad \text{MMD}_{\text{OPT}} \triangleq \text{MMD}_{\boldsymbol{k}_{\mathbb{P}}}(\textstyle\sum_{i=1}^n w_{\text{OPT}\,i} \delta_{x_i}, \mathbb{P}). \end{aligned} \tag{3.3}$$

For example, Hodgkinson, Salomone, and Roosta [HSR20, Thm. 1b] showed that simplex reweighting can correct for biases due to off-target i.i.d. or MCMC sampling. Our next result (proved in Sec. 3.C.2) significantly relaxes their conditions.

Theorem 3.1 (Debiasing via simplex reweighting)**.** *Consider a kernel $\boldsymbol{k}_{\mathbb{P}}$ satisfying Assum. 3.1 with $\mathcal{H}_{\boldsymbol{k}_{\mathbb{P}}}$ separable, and suppose $(x_i)_{i=1}^{\infty}$ are the iterates of a homogeneous ϕ-irreducible geometrically ergodic Markov chain [GLR23, Thm. 1] with stationary distribution $\mathbb{Q}$ and initial distribution absolutely continuous with respect to $\mathbb{P}$. If*

$$\mathbb{E}_{x\sim\mathbb{P}}[\tfrac{\mathrm{d}\mathbb{P}}{\mathrm{d}\mathbb{Q}}(x)^{2\mathfrak{q}-1}\boldsymbol{k}_{\mathbb{P}}(x,x)^{\mathfrak{q}}] < \infty$$

for some $\mathfrak{q} > 1$ then $\mathrm{MMD}_{\mathrm{OPT}} = O(n^{-1/2})$ in probability.

Remark 3.1. *$\mathcal{H}_{\boldsymbol{k}_{\mathbb{P}}}$ is separable whenever $\boldsymbol{k}_{\mathbb{P}}$ is continuous [SC08, Lem. 4.33].*

Since n points sampled i.i.d. from $\mathbb{P}$ have $\Theta(n^{-1/2})$ root mean squared MMD (see Prop. 3.C.1), Thm. 3.1 shows that a debiased off-target sample can be as accurate as a direct sample from $\mathbb{P}$. Moreover, Thm. 3.1 applies to many practical examples. The simplest example of a geometrically ergodic chain is i.i.d. sampling from $\mathbb{Q}$, but geometric ergodicity has also been established for a variety of popular Markov chains including random walk Metropolis [RT96, Thm. 3.2], independent Metropolis-Hastings [AP07, Thm. 2.2], the unadjusted Langevin algorithm [DM17, Prop. 8], the Metropolis-adjusted Langevin algorithm [DM22a, Thm. 1], Hamiltonian Monte Carlo [DMS20, Thm. 10 and Thm. 11], stochastic gradient Langevin dynamics [LLW23, Thm. 2.1], and the Gibbs sampler [Joh09]. Moreover, for $\mathbb{Q}$ absolutely continuous with respect to $\mathbb{P}$, the importance weight $\frac{\mathrm{d}\mathbb{P}}{\mathrm{d}\mathbb{Q}}$ is typically bounded or slowly growing when the tails of $\mathbb{Q}$ are not much lighter than those of $\mathbb{P}$.

Remarkably, under more stringent conditions, Thm. 3.2 (proved in Sec. 3.C.3) shows that simplex reweighting can decrease MMD to $\mathbb{P}$ at an even-faster-than-i.i.d. rate.

Theorem 3.2 (Better-than-i.i.d. debiasing via simplex reweighting)**.** *Consider a kernel $\boldsymbol{k}_{\mathbb{P}}$ satisfying Assum. 3.1 with $\mathfrak{p} = 2$ and points $(x_i)_{i=1}^{\infty}$ drawn i.i.d. from a distribution $\mathbb{Q}$ with $\frac{\mathrm{d}\mathbb{P}}{\mathrm{d}\mathbb{Q}}$ bounded. If $\mathbb{E}[\boldsymbol{k}_{\mathbb{P}}(x_1,x_1)^{\mathfrak{q}}] < \infty$ for some $\mathfrak{q} > 3$, then $\mathbb{E}[\mathrm{MMD}^2_{\mathrm{OPT}}] = o(n^{-1})$.*

The work of Liu and Lee [LL17, Thm. 3.3] also established $o(n^{-1/2})$ MMD error for simplex reweighting but only under a uniformly bounded eigenfunctions assumption that is often violated ([Min10, Thm. 1], [Zho02, Ex. 1]) and difficult to verify [SS12].

Our remaining results make no particular assumption about the input points but rather upper bound the excess MMD

$$\Delta\mathrm{MMD}_{\boldsymbol{k}_{\mathbb{P}}}(w) \triangleq \mathrm{MMD}_{\boldsymbol{k}_{\mathbb{P}}}(\textstyle\sum_{i\in[n]} w_i\delta_{x_i}, \mathbb{P}) \; - \mathrm{MMD}_{\mathrm{OPT}} \tag{3.4}$$

of a candidate weighting w in terms of the input point radius $R_n \triangleq \max_{i\in[n]} \|x_i\|_2 \vee 1$ and kernel radius $\|\boldsymbol{k}_{\mathbb{P}}\|_n \triangleq \max_{i\in[n]} \boldsymbol{k}_{\mathbb{P}}(x_i, x_i)$. While these results apply to *any* input points, we will consider the following running example of *slow-growing* input points throughout the paper.

Definition 3.3 (Slow-growing input points)**.** *We say* $\mathcal{S}_n$ *is* γ-slow-growing *if* $R_n = O((\log n)^\gamma)$ *for some* $\gamma \geq 0$ *and* $\|\boldsymbol{k}_{\mathbb{P}}\|_n = \widetilde{O}(1)$.

Notably, $\mathcal{S}_n$ is 1-slow-growing with probability 1 when $\boldsymbol{k}_{\mathbb{P}}(x, x)$ is polynomially bounded by $\|x\|_2$ and the input points are drawn from a homogeneous ϕ-irreducible geometrically ergodic Markov chain with a sub-exponential target $\mathbb{Q}$, i.e., $\mathbb{E}[e^{c\|x\|_2}] < \infty$ for some $c > 0$ [DM21, Prop. 2]. For a Stein kernel $\boldsymbol{k}_p$ (Def. 3.1), by Prop. 3.B.3, $\boldsymbol{k}_p(x, x)$ is polynomially bounded by $\|x\|_2$ if $\boldsymbol{k}(x, x)$, $\|\nabla_x \nabla_y \boldsymbol{k}(x, x)\|_2$, and $\|\nabla \log p(x)\|_2$ are all polynomially bounded by $\|x\|_2$. Moreover, $\|\nabla \log p(x)\|_2$ is automatically polynomially bounded by $\|x\|_2$ when $\nabla \log p$ is Lipschitz or, more generally, pseudo-Lipschitz [EMS18, Eq. (2.5)].

♢ 3.2.3 Debiased compression via Stein Kernel Thinning

Off-the-shelf solvers based on mirror descent and Frank Wolfe can solve the convex debiasing program (3.3) in $O(n^3)$ time by generating weights with $O(n^{-1/2}\|\boldsymbol{k}_{\mathbb{P}}\|_n)$ excess MMD [LL17]. We instead employ a more efficient, greedy debiasing strategy based on Stein thinning (ST). After n rounds, ST outputs an equal-weighted coreset of size n with $O(n^{-1/2}\|\boldsymbol{k}_{\mathbb{P}}\|_n)$ excess MMD [Ria+22, Thm. 1]. Moreover, while the original implementation of Riabiz et al. [Ria+22] has cubic runtime, our implementation (Alg. 3.D.1) based on sufficient statistics improves the runtime to $O(n^2 d_{\boldsymbol{k}_{\mathbb{P}}})$ where $d_{\boldsymbol{k}_{\mathbb{P}}}$ denotes the runtime of a single kernel evaluation.[3]

The equal-weighted output of ST serves as the perfect input for the kernel thinning (KT) algorithm which compresses an equal-weighted sample of size n into a coreset of any target size $m \leq n$ in $O(n^2 d_{\boldsymbol{k}_{\mathbb{P}}})$ time. We adapt the target KT algorithm slightly to target MMD error to $\mathbb{P}$ and to include a baseline ST coreset of size m in the KT-SWAP step (see Alg. 3.D.3). Combining the two routines we obtain Stein Kernel Thinning (SKT), our first solution for equal-weighted debiased distribution compression:

Our next result, proved in Sec. 3.D.3, shows that SKT yields better-than-i.i.d. excess MMD whenever the radii (R_n and $\|\boldsymbol{k}_{\mathbb{P}}\|_n$) and kernel covering number exhibit slow growth.

Theorem 3.3 (MMD guarantee for SKT)**.** *Given a kernel* $\boldsymbol{k}_{\mathbb{P}}$ *satisfying Assums. 3.1 and* $(\boldsymbol{\alpha}, \boldsymbol{\beta})$*-kernel, Stein Kernel Thinning (Alg. 3.2) outputs* w_{SKT} *in* $O(n^2 d_{\boldsymbol{k}_{\mathbb{P}}})$ *time satisfying*

$$\Delta\mathrm{MMD}_{\boldsymbol{k}_{\mathbb{P}}}(w_{\mathrm{SKT}}) = O\Big(\frac{\sqrt{\|\boldsymbol{k}_{\mathbb{P}}\|_n \ell_\delta \cdot \log n \cdot R_n^\beta G_m^\alpha}}{\min(m, \sqrt{n})}\Big)$$

[3] Often, $d_{\boldsymbol{k}_{\mathbb{P}}} = \Theta(d)$ as in the case of Stein kernels (Sec. 3.I.1).

Algorithm 3.2 Stein Kernel Thinning (SKT)

Input: mean-zero kernel $\boldsymbol{k}_{\mathbb{P}}$, points $\mathcal{S}_n$, output size m, KT failure probability δ
$n' \leftarrow m\, 2^{\lceil \log_2 \frac{n}{m} \rceil}$
$w \leftarrow \text{SteinThinning}(\boldsymbol{k}_{\mathbb{P}}, \mathcal{S}_n, n')$
$w_{\text{SKT}} \leftarrow \text{KernelThinning}(\boldsymbol{k}_{\mathbb{P}}, \mathcal{S}_n, n', w, m, \delta)$
Return: $w_{\text{SKT}} \in \Delta_{n-1} \cap (\frac{\mathbb{N}_0}{m})^n$ ▷ *hence* $\|w_{\text{SKT}}\|_0 \leq m$

with probability at least $1-\delta$, *where* $\ell_\delta \triangleq \log^2(\frac{e}{\delta})$ *and*

$$G_m \triangleq \begin{cases} \log(em) & \textsc{LogGrowth}(\alpha, \beta), \\ m & \textsc{PolyGrowth}(\alpha, \beta). \end{cases}$$

Example 3.1. *Under the assumptions of Thm. 3.3 with γ-slow-growing input points (Def. 3.3),* LogGrowth $\boldsymbol{k}_{\mathbb{P}}$, *and a coreset size $m \leq \sqrt{n}$,* SKT *delivers $\widetilde{O}(m^{-1})$ excess MMD with high probability, a significant improvement over the $\Omega(m^{-1/2})$ error rate of i.i.d. sampling.*

Remark 3.2. *When $m < \sqrt{n}$, we can uniformly subsample or, in the case of MCMC inputs,* standard thin *(i.e., keep only every $\frac{n}{m^2}$-th point of) the input sequence down to size m^2 before running* SKT *to reduce runtime while incurring only $O(m^{-1})$ excess error. The same holds for the* LSKT *algorithm introduced in Sec. 3.3.*

■ 3.3 Accelerated Debiased Compression

To enable larger-scale debiased compression, we next introduce a sub-quadratic-time version of SKT built via a new low-rank debiasing scheme and the near-linear-time compression algorithm of Shetty, Dwivedi, and Mackey [SDM22].

◇ 3.3.1 Fast bias correction via low-rank approximation

At a high level, our approach to accelerated debiasing involves four components. First, we form a rank-r approximation $FF^\top$ of the kernel matrix $K = \boldsymbol{k}_{\mathbb{P}}(\mathcal{S}_n, \mathcal{S}_n)$ in $O(nrd_{\boldsymbol{k}_{\mathbb{P}}} + nr^2)$ time using a weighted extension (WeightedRPCholesky, Alg. 3.F.1) of the randomly pivoted Cholesky algorithm of Chen et al. [Che+22b, Alg. 2.1]. Second, we correct the diagonal to form $K' = FF^\top + \text{diag}(K - FF^\top)$. Third, we solve the reweighting problem (3.3) with K' substituted for K using T iterations of accelerated entropic mirror descent [AMD, WAL23, Alg. 14 with $\phi(w) = \sum_i w_i \log w_i$]. The acceleration ensures $O(1/T^2)$ suboptimality after T iterations, and each iteration takes only $O(nr)$ time thanks to the low-rank plus diagonal approximation. Finally, we repeat this three-step procedure Q times, each

time using the weights outputted by the prior round to update the low-rank approximation $\widehat{K}$. On these subsequent adaptive rounds, WeightedRPCholesky approximates the leading subspace of a *weighted* kernel matrix $\mathrm{diag}(\sqrt{\tilde{w}})K\,\mathrm{diag}(\sqrt{\tilde{w}})$ before undoing the row and column reweighting. Since each round's weights are closer to optimal, this adaptive updating has the effect of upweighting more relevant subspaces for subsequent debiasing. For added sparsity, we prune the weights outputted by the prior round using stratified residual resampling [Resample, Alg. 3.E.3, DC05]. Our complete Low-rank Debiasing (LD) scheme, summarized in Alg. 3.3, enjoys $o(n^2)$ runtime whenever $r = o(n^{1/2})$, $T = O(n^{1/2})$, and $Q = O(1)$.

Algorithm 3.3 Low-rank Debiasing (LD)

Input: mean-zero kernel $\boldsymbol{k}_{\mathbb{P}}$, points $\mathcal{S}_n = (x_i)_{i=1}^n$, rank r, AMD steps T, adaptive rounds Q
$w^{(0)} \leftarrow (\frac{1}{n}, \dots, \frac{1}{n}) \in \mathbb{R}^n$
for $q = 1$ **to** Q **do**
  $\tilde{w} \leftarrow \text{Resample}(w^{(q-1)}, n)$
  $\mathtt{I}, F \leftarrow \text{WeightedRPCholesky}(\boldsymbol{k}_{\mathbb{P}}, \mathcal{S}_n, \tilde{w}, r)$
  $K' \leftarrow FF^\top + \mathrm{diag}(\boldsymbol{k}_{\mathbb{P}}(\mathcal{S}_n, \mathcal{S}_n)) - \mathrm{diag}(FF^\top)$
  $w^{(q)} \leftarrow \text{AMD}(K', T, \tilde{w}, \texttt{AGG} = \mathbf{1}_{q>1})$
  if $(w^{(q)})^\top K' w^{(q)} > \tilde{w}^\top K' \tilde{w}$ **then** $w^{(q)} \leftarrow \tilde{w}$
end for
Return: $w_{\mathrm{LD}} \leftarrow w^{(Q)} \in \Delta_{n-1}$

Moreover, our next result, proved in Sec. 3.F.1, shows that LD provides i.i.d.-level precision whenever $T \geq \sqrt{n}$, $Q = O(1)$, and r grows appropriately with the input radius and kernel covering number.

Assumption ($\boldsymbol{\alpha}$,$\boldsymbol{\beta}$)-params. *The kernel $\boldsymbol{k}_{\mathbb{P}}$ satisfies Assums. 3.1 and ($\boldsymbol{\alpha}$, $\boldsymbol{\beta}$)-kernel, the output size and rank $m, r \geq (\frac{\mathfrak{C}_d R_n^\beta + 1}{\sqrt{\log 2}} + 2\sqrt{\log 2})^2$, the AMD step count $T \geq \sqrt{n}$, and the adaptive round count $Q = O(1)$.*[4]

Theorem 3.4 (Debiasing guarantee for LD)**.** *Under Assum. ($\boldsymbol{\alpha}$,$\boldsymbol{\beta}$)-params, Low-rank Debiasing (Alg. 3.3) takes $O((d_{\boldsymbol{k}_{\mathbb{P}}} + r + T)nr)$ time to output w_{LD} satisfying*

$$\Delta\mathrm{MMD}_{\boldsymbol{k}_{\mathbb{P}}}(w_{\mathrm{LD}}) = O\left(\sqrt{\frac{\|\boldsymbol{k}_{\mathbb{P}}\|_n \max(\log n, 1/\delta)}{n}} + \sqrt{\frac{nH_{n,r}}{\delta}}\right)$$

[4] To unify the presentation of our results, Assum. ($\boldsymbol{\alpha}$,$\boldsymbol{\beta}$)-params constrains all common algorithm input parameters with the understanding that the conditions are enforced only when the input is relevant to a given algorithm.

with probability at least $1-\delta$, *for any* $\delta \in (0,1)$ *and* $H_{n,r}$ *defined in* (3.48) *that satisfies*

$$H_{n,r} = \begin{cases} O\left(\sqrt{r}\left(\frac{R_n^{2\beta}}{r}\right)^{\frac{1}{\alpha}}\right) & \textsc{PolyGrowth}(\alpha,\beta), \\ O\left(\sqrt{r}\exp\left(-\left(\frac{0.83\sqrt{r}-2.39}{\mathfrak{C}_d R_n^{\beta}}\right)^{\frac{1}{\alpha}}\right)\right) & \textsc{LogGrowth}(\alpha,\beta). \end{cases}$$

Example 3.2. *Under the assumptions of Thm. 3.4 with* γ*-slow-growing input points (Def. 3.3),* LogGrowth $\boldsymbol{k}_{\mathbb{P}}$, $T = \Theta(\sqrt{n})$, *and* $r = (\log n)^{2(\alpha+\beta\gamma)+\epsilon}$ *for any* $\epsilon > 0$, LD *delivers* $\widetilde{O}(n^{-1/2})$ *excess MMD with high probability in* $\widetilde{O}(n^{1.5})$ *time.*

♢ 3.3.2 Fast debiased compression via Low-rank Stein KT

To achieve debiased compression in sub-quadratic time, we next propose Low-rank SKT (Alg. 3.4). LSKT debiases the input using LD, converts the LD output into an equal-weighted coreset using Resample, and finally combines KT with the divide-and-conquer Compress++ framework [SDM22] to compress n equal-weighted points into $\sqrt{n}$ in near-linear time.

Algorithm 3.4 Low-rank Stein Kernel Thinning (LSKT)

Input: mean-zero kernel $\boldsymbol{k}_{\mathbb{P}}$, points $\mathcal{S}_n = (x_i)_{i=1}^n$, rank r, AGM steps T, adaptive rounds Q, oversampling parameter $\mathfrak{g}$, failure prob. δ
$w \leftarrow$ Low-rankDebiasing$(\boldsymbol{k}_{\mathbb{P}}, \mathcal{S}_n, r, T, Q)$
$n' \leftarrow 4^{\lceil \log_4 n \rceil}, m \leftarrow \sqrt{n'}$ ▷ *output size* $\sqrt{n} \le m < 2\sqrt{n}$
$w \leftarrow$ Resample(w, n')
$w_{\text{LSKT}} \leftarrow$ KT-Compress++$(\boldsymbol{k}_{\mathbb{P}}, \mathcal{S}_n, n', w, \mathfrak{g}, \frac{\delta}{3})$
Return: $w_{\text{LSKT}} \in \Delta_{n-1} \cap (\frac{\mathbb{N}_0}{m})^n$ ▷ *hence* $\|w_{\text{LSKT}}\|_0 \le m$

Our next result (proved in Sec. 3.F) shows that LSKT can provide better-than-i.i.d. excess MMD in $o(n^2)$ time.

Theorem 3.5 (MMD guarantee for LSKT)**.** *Under Assum.* $(\boldsymbol{\alpha},\boldsymbol{\beta})$*-params, Low-rank SKT (Alg. 3.4) with* $\mathfrak{g} \in [\log_2 \log(n+1) + 3.1, \log_4(\sqrt{n}/\log n)]$ *and* $\delta \in (0,1)$ *outputs* w_{LSKT} *in* $O((d_{\boldsymbol{k}_{\mathbb{P}}}+r+T)nr + d_{\boldsymbol{k}_{\mathbb{P}}} n^{1.5})$ *time satisfying, with probability at least* $1-\delta$,

$$\Delta\text{MMD}_{\boldsymbol{k}_{\mathbb{P}}}(w_{\text{LSKT}}) = O\left(\sqrt{\frac{\|\boldsymbol{k}_{\mathbb{P}}\|_n \max(1/\delta, \ell_\delta(\log n) n^{\gamma\beta} G^{\alpha}_{\sqrt{n}})}{n}} + \sqrt{\frac{nH_{n,r}}{\delta}}\right),$$

for G_m, $H_{n,r}$ *as in Thms. 3.3 and 3.5.*

Example 3.3. *Under the assumptions of Thm. 3.5 with* γ*-slow-growing input points (Def. 3.3),* LogGrowth $\boldsymbol{k}_{\mathbb{P}}$, $T = \Theta(\sqrt{n})$, *and* $r = (\log n)^{2(\alpha+\beta\gamma)+\epsilon}$ *for any* $\epsilon > 0$, LSKT *delivers* $\widetilde{O}(n^{-1/2})$ *excess MMD with high probability in* $\widetilde{O}(n^{1.5})$ *time with a coreset of size* $m \in [\sqrt{n}, 2\sqrt{n})$.

■ 3.4 Weighted Debiased Compression

The prior sections developed debiased equal-weighted coresets with better-than-i.i.d. compression guarantees. In this section, we match those guarantees with significantly smaller weighted coresets.

◇ 3.4.1 Simplex-weighted coresets via Stein Recombination

Algorithm 3.5 Recombination Thinning (RT)

Input: mean-zero kernel $\boldsymbol{k}_{\mathbb{P}}$, points $\mathcal{S}_n = (x_i)_{i=1}^n$, weights $w \in \Delta_{n-1}$, output size m
$\tilde{w} \leftarrow$ Resample(w, n)
$\mathtt{I}, F \leftarrow$ WeightedRPCholesky$(\boldsymbol{k}_{\mathbb{P}}, \mathcal{S}_n, \tilde{w}, m-1)$
$w' \leftarrow$ Recombination$([F, \mathbf{1}_n]^\top, \tilde{w})$ ▷ $[F, \mathbf{1}_n]^\top \in \mathbb{R}^{m \times n}$
▷ $F^\top \tilde{w} = F^\top w'$, $w' \in \Delta_{n-1}$, *and* $\|w'\|_0 \leq m$
$w'' \leftarrow$ KT-Swap-LS$(\boldsymbol{k}_{\mathbb{P}}, \mathcal{S}_n, w', \text{SPLX})$; $\mathtt{J} \leftarrow \{i : w''_i > 0\}$
$w''[\mathtt{J}] \leftarrow \arg\min_{w' \in \Delta_{|\mathtt{J}|-1}} w'^\top \boldsymbol{k}_{\mathbb{P}}(\mathcal{S}_n[\mathtt{J}], \mathcal{S}_n[\mathtt{J}])w'$ ▷ *use any* $O(|\mathtt{J}|^3)$ *quadratic programming solver*
Return: $w_{\mathrm{RT}} \leftarrow w'' \in \Delta_{n-1}$ with $\|w_{\mathrm{RT}}\|_0 \leq m$

Inspired by the coreset constructions of Hayakawa, Oberhauser, and Lyons [HOL22; HOL23], we first introduce a simplex-weighted compression algorithm, Recombination-Thinning (RT, Alg. 3.5), suitable for summarizing a debiased input sequence. To produce a coreset given input weights $w \in \Delta_{n-1}$, RT first prunes small weights using Resample and then uses WeightedRPCholesky to identify $m-1$ test vectors that capture most of the variability in the weighted kernel matrix. Next, Recombination (Alg. 3.G.1) [Tch16, Alg. 1] identifies a sparse simplex vector w' with $\|w'\|_0 \leq m$ that exactly matches the inner product of its input with each of the test vectors. Then, we run KT-Swap-LS (Alg. 3.G.2), a new, line-search version of KT-SWAP [DM21, Alg. 1b] that greedily improves MMD to $\mathbb{P}$ while maintaining both the sparsity and simplex constraint of its input. Finally, we optimize the weights of the remaining support points using any cubic-time quadratic programming solver.

In Prop. 3.G.1 we show that RT runs in time $O((d_{\boldsymbol{k}_{\mathbb{P}}} + m)nm + m^3 \log n)$ and nearly preserves the MMD of its input whenever m grows appropriately with the kernel covering number. Combining RT with SteinThinning or Low-rankDebiasing in Alg. 3.6, we obtain Stein Recombination (SR) and Low-rank SR (LSR), our approaches to debiased simplex-weighted compression. Remarkably, SR and LSR can match the MMD error rates established for SKT and LSKT using substantially fewer coreset points, as our next result (proved in Sec. 3.G.2) shows.

Algorithm 3.6 (Low-rank) Stein Recombination (SR / LSR)

Input: mean-zero kernel $\boldsymbol{k}_{\mathbb{P}}$, points $\mathcal{S}_n$, output size m, rank r, AGM steps T, adaptive rounds Q

$$w \leftarrow \begin{cases} \text{Low-rankDebiasing}(\boldsymbol{k}_{\mathbb{P}}, \mathcal{S}_n, r, T, Q) & \text{if low-rank} \\ \text{SteinThinning}(\boldsymbol{k}_{\mathbb{P}}, \mathcal{S}_n) & \text{otherwise} \end{cases}$$

$w_{\mathrm{SR}} \leftarrow \text{RecombinationThinning}(\boldsymbol{k}_{\mathbb{P}}, \mathcal{S}_n, w, m)$

Return: $w_{\mathrm{SR}} \in \Delta_{n-1}$ with $\|w_{\mathrm{SR}}\|_0 \leq m$

Theorem 3.6 (MMD guarantee for SR/ LSR)**.** *Under Assum.* $(\boldsymbol{\alpha},\boldsymbol{\beta})$*-params, Stein Recombination (Alg. 3.6) takes* $O(d_{\boldsymbol{k}_{\mathbb{P}}} n^2 + (d_{\boldsymbol{k}_{\mathbb{P}}} + m)nm + m^3 \log n)$ *to output* w_{SR}*, and Low-rank SR takes* $O((d_{\boldsymbol{k}_{\mathbb{P}}} + r + T)nr + (d_{\boldsymbol{k}_{\mathbb{P}}} + m)nm + m^3 \log n)$ *time to output* w_{LSR}*. Moreover, for any* $\delta \in (0,1)$ *and* $H_{n,r}$ *as in Thm. 3.4, each of the following bounds holds (separately) with probability at least* $1-\delta$*:*

$$\Delta\mathrm{MMD}_{\boldsymbol{k}_{\mathbb{P}}}(w_{\mathrm{SR}}) = O\left(\sqrt{\frac{\|\boldsymbol{k}_{\mathbb{P}}\|_n (\log n \vee \frac{1}{\delta})}{n} + \frac{nH_{n,m}}{\delta}}\right) \text{ and}$$

$$\Delta\mathrm{MMD}_{\boldsymbol{k}_{\mathbb{P}}}(w_{\mathrm{LSR}}) = O\left(\sqrt{\frac{\|\boldsymbol{k}_{\mathbb{P}}\|_n (\log n \vee \frac{1}{\delta})}{n} + \frac{n(H_{n,m}+H_{n,r})}{\delta}}\right).$$

Example 3.4. *Instantiate the assumptions of Thm. 3.6 with* γ*-slow-growing input points (Def. 3.3),* LogGrowth $\boldsymbol{k}_{\mathbb{P}}$*, and a heavily compressed coreset size* $m = (\log n)^{2(\alpha+\beta\gamma)+\epsilon}$ *for any* $\epsilon > 0$*. Then SR delivers* $\widetilde{O}(n^{-1/2})$ *excess MMD with high probability in* $O(n^2)$ *time, and LSR with* $r = m$ *and* $T = \Theta(\sqrt{n})$ *achieves the same in* $\widetilde{O}(n^{1.5})$ *time.*

♢ 3.4.2 Constant-preserving coresets via Stein Cholesky

For applications supporting negative weights, we next introduce a constant-preserving compression algorithm, CholeskyThinning (CT, Alg. 3.7), suitable for summarizing a debiased input sequence. CT first applies WeightedRPCholesky to a *constant-regularized kernel* $\boldsymbol{k}_{\mathbb{P}}(x,y) + c$ to select an initial coreset and then uses a combination of KT-Swap-LS and closed-form optimal constant-preserving reweighting to greedily refine the support and weights. The regularized kernel ensures that WeightedRPCholesky, originally developed for compression with unconstrained weights, also yields a high-quality coreset when its weights are constrained to sum to 1, and our CT standalone analysis (Prop. 3.H.1) improves upon the runtime and error guarantees of RT. In Alg. 3.8, we combine CT with SteinThinning or Low-rankDebiasing to obtain Stein Cholesky (SC) and Low-rank SC (LSC), our approaches to debiased constant-preserving compression. Our MMD guarantees for SC and LSC (proved in Sec. 3.H.2) improve upon the rates of Thm. 3.6.

Algorithm 3.7 Cholesky Thinning (CT)

Input: mean-zero kernel $\boldsymbol{k}_{\mathbb{P}}$, points $\mathcal{S}_n = (x_i)_{i=1}^n$, weights $w \in \Delta_{n-1}$, output size m
$c \leftarrow$ AVERAGE(Largest m entries of $(\boldsymbol{k}_{\mathbb{P}}(x_i, x_i))_{i=1}^n$)
$\mathtt{I}, F \leftarrow$ WeightedRPCholesky$(\boldsymbol{k}_{\mathbb{P}} + c, \mathcal{S}_n, w, m)$; $w \leftarrow \mathbf{0}_n$
$w[\mathtt{I}] \leftarrow \arg\min_{w' \in \mathbb{R}^{|\mathtt{I}|}: \sum_i w'_i = 1} w'^{\top} \boldsymbol{k}_{\mathbb{P}}(\mathcal{S}_n[\mathtt{I}], \mathcal{S}_n[\mathtt{I}]) w'$
$w \leftarrow$ KT-Swap-LS$(\boldsymbol{k}_{\mathbb{P}}, \mathcal{S}_n, w, \mathrm{CP})$; $\mathtt{I} \leftarrow \{i : w_i \neq 0\}$
$w[\mathtt{I}] \leftarrow \arg\min_{w' \in \mathbb{R}^{|\mathtt{I}|}: \sum_i w'_i = 1} w'^{\top} \boldsymbol{k}_{\mathbb{P}}(\mathcal{S}_n[\mathtt{I}], \mathcal{S}_n[\mathtt{I}]) w'$
Return: $w_{\mathrm{CT}} \leftarrow w \in \mathbb{R}^n$ with $\|w_{\mathrm{CT}}\|_0 \leq m$, $\mathbf{1}_n^{\top} w_{\mathrm{CT}} = 1$

Algorithm 3.8 (Low-rank) Stein Cholesky (SC / LSC)

Input: mean-zero kernel $\boldsymbol{k}_{\mathbb{P}}$, points $\mathcal{S}_n$, output size m, rank r, AGM steps T, adaptive rounds Q

$$w \leftarrow \begin{cases} \text{Low-rankDebiasing}(\boldsymbol{k}_{\mathbb{P}}, \mathcal{S}_n, r, T, Q) & \text{if low-rank} \\ \text{SteinThinning}(\boldsymbol{k}_{\mathbb{P}}, \mathcal{S}_n) & \text{otherwise} \end{cases}$$

$w_{\mathrm{SC}} \leftarrow$ CholeskyThinning$(\boldsymbol{k}_{\mathbb{P}}, \mathcal{S}_n, w, m)$
Return: $w_{\mathrm{SC}} \in \mathbb{R}^n$ with $\|w_{\mathrm{SC}}\|_0 \leq m$ and $\mathbf{1}_n^{\top} w_{\mathrm{SC}} = 1$

Theorem 3.7 (MMD guarantee for SC / LSC)**.** *Under Assum. ($\boldsymbol{\alpha}$,$\boldsymbol{\beta}$)-params, Stein Cholesky (Alg. 3.8) takes $O(d_{\boldsymbol{k}_{\mathbb{P}}} n^2 + (d_{\boldsymbol{k}_{\mathbb{P}}} + m) nm + m^3)$ time to output w_{SC}, and Low-rank SC takes $O((d_{\boldsymbol{k}_{\mathbb{P}}} + r + T) nr + (d_{\boldsymbol{k}_{\mathbb{P}}} + m) nm + m^3)$ time to output w_{LSC}. Moreover, for any $\delta \in (0, 1)$, with probability at least $1 - \delta$, each of the following bounds hold:*

$$\Delta\mathrm{MMD}_{\boldsymbol{k}_{\mathbb{P}}}(w_{\mathrm{SC}}) = 2\,\mathrm{MMD}_{\mathrm{OPT}} + O\Big(\sqrt{\tfrac{\|\boldsymbol{k}_{\mathbb{P}}\|_n \log n}{\delta n} + \tfrac{H_{n,m'}}{\delta}}\Big) \text{ and}$$
$$\Delta\mathrm{MMD}_{\boldsymbol{k}_{\mathbb{P}}}(w_{\mathrm{LSC}}) = 2\,\mathrm{MMD}_{\mathrm{OPT}} + O\Big(\sqrt{\tfrac{\|\boldsymbol{k}_{\mathbb{P}}\|_n (\log n \vee 1/\delta)}{\delta n} + \tfrac{H_{n,m'}}{\delta} + \tfrac{n H_{n,r}}{\delta^2}}\Big)$$

for $H_{n,r}$ as in Thm. 3.4 and $m' \triangleq m + \log 2 - 2\sqrt{m \log 2 + 1}$.

Example 3.5. *Instantiate the assumptions of Thm. 3.7 with γ-slow-growing input points (Def. 3.3),* LOGGROWTH $\boldsymbol{k}_{\mathbb{P}}$*, and a heavily compressed coreset size $m = (\log n)^{2(\alpha+\beta\gamma)+\epsilon}$ for any $\epsilon > 0$. Then SC delivers $\widetilde{O}(n^{-1/2})$ excess MMD with high probability in $O(n^2)$ time, and LSC with $r = m$ and $T = \Theta(\sqrt{n})$ achieves the same in $\widetilde{O}(n^{1.5})$ time.*

Remark 3.3. *While we present our results for a target precision of $1/\sqrt{n}$, a coarser target precision of $1/\sqrt{n_0}$ for $n_0 < n$ can be achieved more quickly by standard thinning the input sequence down to size n_0 before running SR, LSR, SC, or LSC.*

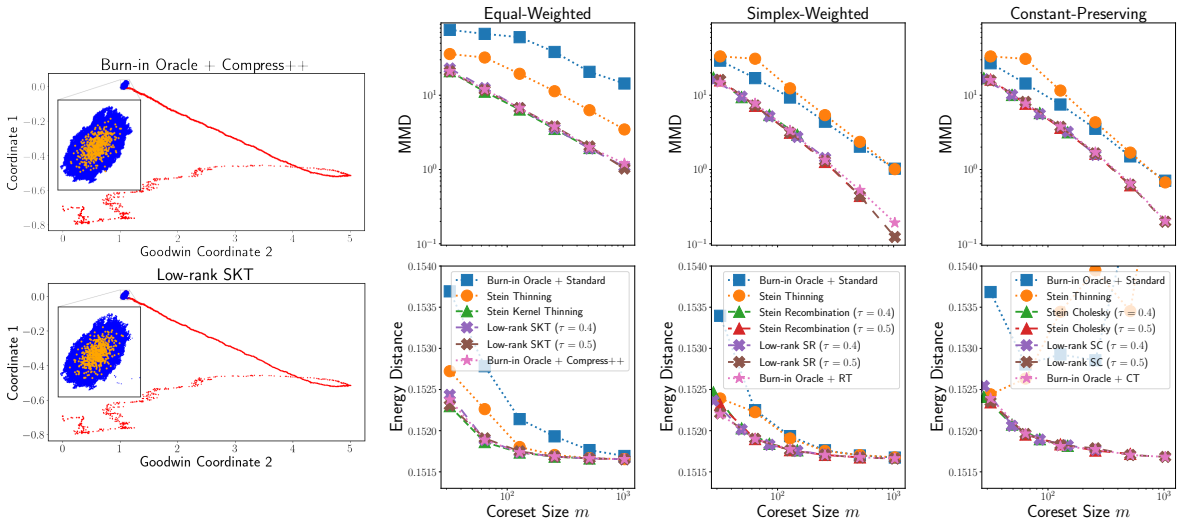

Figure 1: **Correcting for burn-in.** *Left:* Before selecting coresets (orange), the burn-in oracle uses 6 independent Markov chains to discard burn-in (red) while LSKT identifies the same high-density region (blue) with 1 chain. *Right:* Using only one chain, our methods consistently outperform the Stein and standard thinning baselines and match the 6-chain oracle.

■ 3.5 Experiments

We next evaluate the practical utility of our procedures when faced with three common sources of bias: (1) burn-in, (2) approximate MCMC, and (3) tempering. In all experiments, we use a Stein kernel $\boldsymbol{k}_p$ with an inverse multiquadric (IMQ) base kernel $\boldsymbol{k}(x,y) = (1+\|x-y\|_M^2/\sigma^2)^{-1/2}$ for σ equal to the median pairwise $\|\cdot\|_M$ distance amongst 1000 points standard thinned from the input. To vary output MMD precision, we first standard thin the input to size $n_0 \in \{2^{10}, 2^{12}, 2^{14}, 2^{16}, 2^{18}, 2^{20}\}$ before applying any method, as discussed in Rems. 3.2 and 3.3. For low-rank or weighted coreset methods, we show results for $m = r = n^\tau$. When comparing weighted coresets, we optimally reweight every coreset. We report the median over 5 independent runs for all error metrics. We implement our algorithms in JAX [Bra+18] and refer the reader to Sec. 3.I for additional experiment details.

Correcting for burn-in The initial iterates of a Markov chain are biased by its starting point and need not accurately reflect the target distribution $\mathbb{P}$. Classical burn-in corrections use convergence diagnostics to detect and discard these iterates but typically require running multiple independent Markov chains [CC96]. Alternatively, our proposed debiased compression methods can be used to correct for burn-in given just a single chain.

We test this claim using an experimental setup from Riabiz et al. [Ria+22, Sec. 4.1]

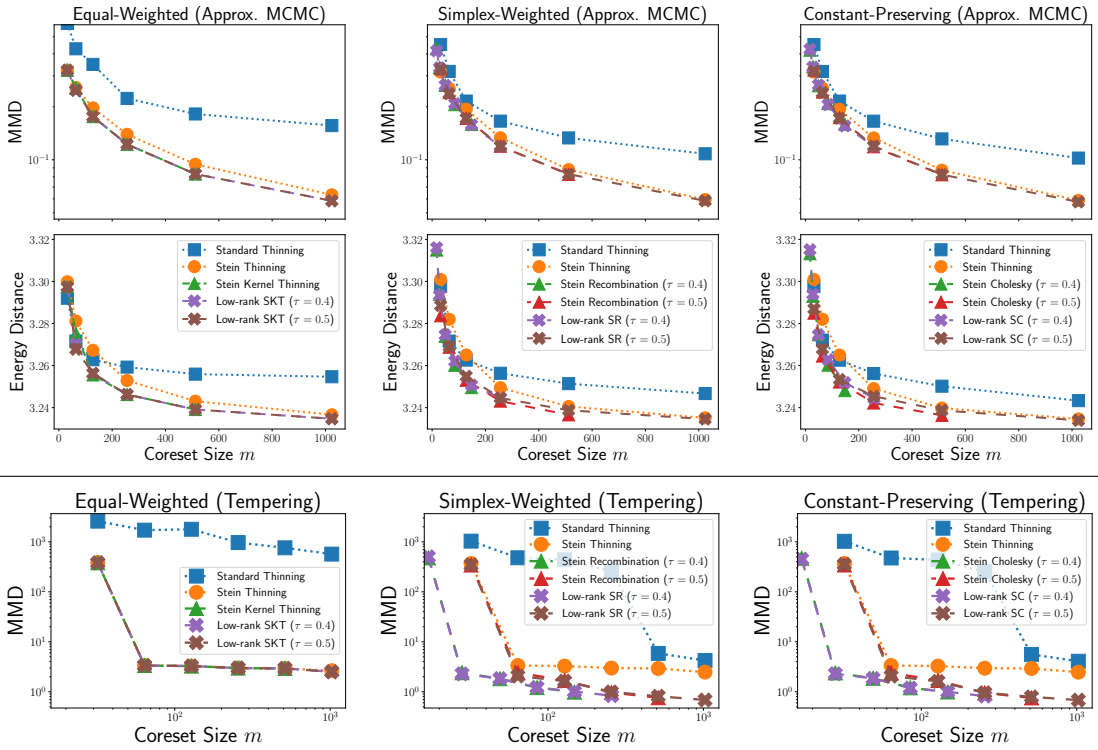

Figure 2: **Correcting for approximate MCMC (top) and tempering (bottom).** For posterior inference over the parameters of Bayesian logistic regression ($d = 54$, top) and a cardiac calcium signaling model ($d = 38$, bottom), our concise coreset constructions correct for approximate MCMC and tempering biases without need for explicit importance sampling.

and the 6-chain "burn-in oracle" diagnostic of Vats and Knudson [VK21]. We aim to compress a posterior $\mathbb{P}$ over the parameters in the Goodwin model of oscillatory enzymatic control ($d = 4$) using $n = 2 \times 10^6$ points from a preconditioned Metropolis-adjusted Langevin algorithm (P-MALA) chain. We repeat this experiment with three alternative MCMC algorithms in Sec. 3.I.3. Our primary metric is $\mathrm{MMD}_{\boldsymbol{k}_{\mathbb{P}}}$ to $\mathbb{P}$ with $M = I$, but, for external validation, we also measure the energy distance [Ria+22, Eq. 11] to an auxiliary MCMC chain of length n. Trajectory plots of the first two coordinates (Fig. 1, left) highlight the substantial burn-in period for the Goodwin chain and the ability of LSKT to mimic the 6-chain burn-in oracle using only a single chain. In Fig. 1 (right), for both the MMD metric and the auxiliary energy distance, our proposed methods consistently outperform Stein thinning and match the quality of 6-chain burn-in removal paired with unbiased compression. The spike in baseline energy distance for the constant-preserving task can be attributed to the selection of overly large weight values due to poor matrix condi-

tioning; the simplex-weighted task does not suffer from this issue due to its regularizing nonnegativity constraint.

Correcting for approximate MCMC In posterior inference, MCMC algorithms typically require iterating over every datapoint to draw each new sample point. When datasets are large, approximating MCMC using datapoint mini-batches can reduce sampling time at the cost of persistent bias and an unknown stationary distribution that prohibits debiasing via importance sampling. Our proposed methods can correct for these biases during compression by computing full-dataset scores on a small subset of n_0 standard thinned points. To evaluate this protocol, we compress a Bayesian logistic regression posterior conditioned on the Forest Covtype dataset ($d = 54$) using $n = 2^{24}$ approximate MCMC points from the stochastic gradient Fisher scoring sampler [AKW12] with batch size 32. Following Wang et al. [Wan+24], we set $M = -\nabla^2 \log p(x_{\text{mode}})$ at the sample mode x_{mode} and use 2^{20} surrogate ground truth points from the No U-turn Sampler [HG+14] to evaluate energy distance. We find that our proposals improve upon standard thinning and Stein thinning for each compression task, not just in the optimized MMD metric (Fig. 2, top) but also in the auxiliary energy distance (Fig. 2, middle) and when measuring integration error for the mean (Fig. 3.I.4).

Correcting for tempering Tempering, targeting a less-peaked and more dispersed distribution $\mathbb{Q}$, is a popular technique to improve the speed of MCMC convergence. One can correct for the sample bias using importance sampling, but this requires knowledge of the tempered density and can introduce substantial variance [GSK10]. Alternatively, one can use constructions of this work to correct for tempering during compression; this requires no importance weighting and no knowledge of $\mathbb{Q}$. To test this proposal, we compress the cardiac calcium signaling model posterior ($d = 38$) of Riabiz et al. [Ria+22, Sec. 4.3] with $M = I$ and $n = 3 \times 10^6$ tempered points from a Gaussian random walk Metropolis-Hastings chain. As discussed by Riabiz et al., compression is essential in this setting as the ultimate aim is to propagate posterior uncertainty through a human heart simulator, a feat which requires over 1000 CPU hours for each summary point retained. Our methods perform on par with Stein thinning for equal-weighted compression and yield substantial gains over Stein (and standard) thinning for the two weighted compression tasks.

■ 3.6 Conclusions and Future Work

We have introduced and analyzed a suite of new procedures for compressing a biased input sequence into an accurate summary of a target distribution. For equal-weighted compression, Stein kernel thinning delivers $\sqrt{n}$ points with $\widetilde{O}(n^{-1/2})$ MMD in $O(n^2)$ time, and low-rank SKT can improve this running time to $\widetilde{O}(n^{3/2})$. For simplex-weighted and constant-preserving compression, Stein recombination and Stein Cholesky provide

enhanced parsimony, matching these guarantees with as few as poly-log(n) points. Recent work has identified some limitations of score-based discrepancies, like Stein kernel MMDs, and developed modified objectives that are more sensitive to the relative density of isolated modes [LDG23; BSD24]. A valuable next step would be to extend our constructions to provide compression guarantees for these modified discrepancy measures. Other opportunities for future work include marrying the better-than-i.i.d. guarantees of this work with the non-myopic compression of Teymur et al. [Tey+21], the control-variate compression of Chopin and Ducrocq [CD21], and the online compression of Hawkins, Koppel, and Zhang [HKZ22].

Appendices

■ 3.A Appendix Notation

For the point sequence $\mathcal{S}_n = (x_i)_{i\in[n]}$, we define $\mathbb{S}_n \triangleq \frac{1}{n}\sum_{i\in[n]}\delta_{x_i}$. For a weight vector $w \in \mathbb{R}^n$, we define the support $\operatorname{supp}(w) \triangleq \{i \in [n] : w_i \neq 0\}$ and the signed measure $\mathbb{S}_n^w \triangleq \sum_{i\in[n]} w_i\delta_{x_i}$. For a matrix $K \in \mathbb{R}^{n\times n}$ and $w \in \Delta_{n-1}$, we define the weighted matrix $K^w \triangleq \operatorname{diag}(\sqrt{w})K\operatorname{diag}(\sqrt{w})$. For positive semidefinite (PSD) matrices (A, B), we use $A \succeq B$ (resp. $A \preceq B$) to mean $A - B$ (resp. $B - A$) is PSD. For a symmetric PSD (SPSD) matrix M, we let $M^{1/2}$ denote a symmetric matrix square root satisfying $M = M^{1/2}M^{1/2}$. For $A \in \mathbb{R}^{n\times m}$, we denote $\|A\|_p \triangleq \sup_{x\neq 0}\frac{\|Ax\|_p}{\|x\|_p}$. We will use $\mathbf{1}_E$ to denote the indicator function for an event E.

Notations used only in a specific section will be introduced within.

■ 3.B Spectral Analysis of Kernel Matrices

The goal of this section is to develop spectral bounds for kernel matrices.

In Sec. 3.B.1, we transfer the bounds on covering numbers from the definition of PolyGrowth or LogGrowth kernels to bounds on the eigenvalues of the kernel matrices. This sets the theoretical foundation for the algorithms in later sections as their error guarantees rely on the fast decay of eigenvalues of kernel matrices.

In Sec. 3.B.2, we show that Stein kernels are PolyGrowth (resp. LogGrowth) provided that their base kernels are differentiable (resp. radially analytic). Hence we obtain spectral bounds for a wide range of Stein kernels.

Notation For a normed space E, we use $\|\cdot\|_E$ to denote its norm, $\mathcal{B}_E(p, r) \triangleq \{x \in E : \|x - p\|_E \leq r\}$ to denote the closed ball of radius r centered at p in E with the shorthand $\mathcal{B}_E(r) \triangleq \mathcal{B}_E(0, r)$ and $\mathcal{B}_E \triangleq \mathcal{B}_E(1)$. When E is an RKHS with kernel $\boldsymbol{k}$, for brevity we use $\boldsymbol{k}$ in place of E in the subscript. Let $\mathfrak{F}(\mathcal{X}, \mathcal{Y})$ denote the space of functions from $\mathcal{X}$ to $\mathcal{Y}$, and $\mathfrak{B}(E, F)$ denote the space of bounded linear functions between normed spaces

E, F. For a set A, we use $\ell_\infty(A)$ to denote the space of bounded $\mathbb{R}$-valued functions on A equipped with the sup-norm $\|f\|_{\infty,A} \triangleq \sup_{x\in A} |f(x)|$. We use $E \hookrightarrow F$ to denote the inclusion map. e use $\lambda_\ell(T)$ to denote the ℓ-th largest eigenvalue of an operator T.

♢ 3.B.1 Bounding the spectrum of kernel matrices

We first introduce the general Mercer representation theorem from Steinwart and Scovel [SS12], which shows the existence of a discrete spectrum of the integral operator associated with a continuous square-integrable kernel. The theorem also provides a series expansion of the kernel, i.e., the Mercer representation, in terms of the eigenvalues and eigenfunctions.

Lemma 3.B.1 (General Mercer representation [SS12])**.** *Consider a kernel $\boldsymbol{k} : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ that is jointly continuous in both inputs and a probability measure μ such that $\int \boldsymbol{k}(x,x)\mathrm{d}\mu(x) < \infty$. Then the following holds.*

(a) The inclusion $\mathcal{H}_{\boldsymbol{k}} \hookrightarrow \mathcal{L}^2(\mu)$ is a compact operator, i.e., $\mathcal{B}_{\boldsymbol{k}}$ is a compact subset of $\mathcal{L}^2(\mu)$. In particular, this inclusion is continuous.

(b) The Hilbert-space adjoint of the inclusion $\mathcal{H}_{\boldsymbol{k}} \hookrightarrow \mathcal{L}^2(\mu)$ is the compact operator $S_{\boldsymbol{k},\mu} : \mathcal{L}^2(\mu) \to \mathcal{H}_{\boldsymbol{k}}$ defined as

$$S_{\boldsymbol{k},\mu} f \triangleq \int \boldsymbol{k}(\cdot, x) f(x) \mathrm{d}\mu(x). \tag{3.5}$$

We also have $S^*_{\boldsymbol{k},\mu} \triangleq \mathcal{H}_{\boldsymbol{k}} \hookrightarrow \mathcal{L}^2(\mu)$. Hence the operator

$$T_{\boldsymbol{k},\mu} \triangleq S^*_{\boldsymbol{k},\mu} S_{\boldsymbol{k},\mu} : \mathcal{L}^2(\mu) \to \mathcal{L}^2(\mu) \tag{3.6}$$

is also compact.

(c) There exist $\{\lambda_\ell\}_{\ell=1}^\infty$ with $\lambda_1 \geq \lambda_2 \geq \cdots \geq 0$ and $\{\phi_\ell\}_{\ell=1}^\infty \subset \mathcal{H}_{\boldsymbol{k}}$ such that $\{\phi_\ell\}_{\ell=1}^\infty$ is an orthonormal system in $\mathcal{L}^2(\mu)$ and $\{\lambda_\ell\}_{\ell=1}^\infty$ (resp. $\{\phi_\ell\}_{\ell=1}^\infty$) consists of the eigenvalues (resp. eigenfunctions) of $T_{\boldsymbol{k},\mu}$ with eigendecomposition, for $f \in \mathcal{L}^2(\mu)$,

$$T_{\boldsymbol{k},\mu} f = \textstyle\sum_{\ell=1}^\infty \lambda_\ell \langle f, \phi_\ell \rangle_{\mathcal{L}^2(\mu)} \phi_\ell$$

with convergence in $\mathcal{L}^2(\mu)$.

(d) We have the following series expansion

$$\boldsymbol{k}(x, x') = \textstyle\sum_{\ell=1}^\infty \lambda_i \phi_\ell(x) \phi_\ell(x'), \tag{3.7}$$

where the series convergence is absolute and uniform in x, x' on all $A \times A \subset \operatorname{supp}\mu \times \operatorname{supp}\mu$.

Proof of Lem. 3.B.1. Part (a) and (b) follow respectively from Steinwart and Scovel [SS12, Lem. 2.3 and 2.2]. Part (c) follows from part (a) and Steinwart and Scovel [SS12, Lem. 2.12]. Finally, part (d) follows from Steinwart and Scovel [SS12, Cor. 3.5]. □

We will use the following lemma regarding the restriction of covering numbers.

Lemma 3.B.2 (Covering number is preserved in restriction)**.** *For a kernel* $\boldsymbol{k} : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ *and a set* $A \subset \mathbb{R}^d$*, we have* $\mathcal{N}_{\boldsymbol{k}}(A, \epsilon) = \mathcal{N}_{\boldsymbol{k}|_A}(A, \epsilon)$*, for* $\boldsymbol{k}|_A$*, the restricted kernel of* $\boldsymbol{k}$ *to* A *[PR16, Sec. 5.4].*

Proof of Lem. 3.B.2. It suffices to show that a $(\boldsymbol{k}, A, \epsilon)$ cover can be converted to a cover of $(\boldsymbol{k}|_A, A, \epsilon)$ of the same cardinality and vice versa.

Let $\mathcal{C} \subset \mathcal{B}_{\boldsymbol{k}|_A}$ be a $(\boldsymbol{k}|_A, A, \epsilon)$ cover. For any $f \in \mathcal{C}$, we have [PR16, Corollary 5.8]

$$\|f\|_{\boldsymbol{k}|_A} = \inf\left\{\|\tilde{f}\|_{\boldsymbol{k}} : \tilde{f} \in \mathcal{H}_{\boldsymbol{k}}, \tilde{f}|_A = f\right\} \leq 1.$$

Moreover, the infimum is attained by some $\tilde{f} \in \mathcal{H}_{\boldsymbol{k}}$ such that $\|\tilde{f}\|_{\boldsymbol{k}} = \|f\|_{\boldsymbol{k}|_A} \leq 1$ and $\tilde{f}|_A = f$. Now form $\tilde{\mathcal{C}} = \{\tilde{f} : f \in \mathcal{C}\}$. For any $\tilde{h} \in \mathcal{B}_{\boldsymbol{k}}$, there exists $f \in \mathcal{C}$ such that

$$\left\|\tilde{h}|_A - f\right\|_{\infty,A} \leq \epsilon \implies = \left\|\tilde{h} - \tilde{f}\right\|_{\infty,A} \leq \epsilon,$$

so $\tilde{\mathcal{C}}$ is a $(\boldsymbol{k}|_A, A, \epsilon)$ cover.

For the other direction, let $\tilde{\mathcal{C}} \subset \mathcal{B}_{\boldsymbol{k}}$ be a $(\boldsymbol{k}, A, \epsilon)$ cover. Define $\mathcal{C} = \{\tilde{f}|_A : \tilde{f} \in \tilde{\mathcal{C}}\} \subset \mathcal{H}_{\boldsymbol{k}|_A}$. Since $\|\tilde{f}|_A\|_{\boldsymbol{k}|_A} \leq \|\tilde{f}\|_{\boldsymbol{k}}$, we have $\mathcal{C} \subset \mathcal{B}_{\boldsymbol{k}|_A}$. For any $h \in \mathcal{B}_{\boldsymbol{k}_A]}$, again by Paulsen and Raghupathi [PR16, Corollary 5.8], there exists $\tilde{h} \in \mathcal{H}_{\boldsymbol{k}}$ such that $\|\tilde{h}\|_{\boldsymbol{k}} = \|h\|_{\boldsymbol{k}|_A} \leq 1$, so there exists $\tilde{f} \in \tilde{\mathcal{C}}$ such that

$$\left\|\tilde{h} - \tilde{f}\right\|_{\infty,A} \leq \epsilon \implies \left\|h - \tilde{f}|_A\right\|_{\infty,A} \leq \epsilon,$$

Hence $\mathcal{C}$ is a $(\boldsymbol{k}, A, \epsilon)$ cover. □

The goal for the rest of this section is to transfer the bounds of the covering number in the definition of a PolyGrowth or LogGrowth kernel from Assum. ($\boldsymbol{\alpha}, \boldsymbol{\beta}$)-kernel to bounds on entropy numbers [SC08, Def. 6.20] that are closely related to eigenvalues of the integral operator (3.6).

Definition 3.B.1 (Entropy number of a bounded linear map)**.** *For a bounded linear operator* $S : E \to F$ *between normed spaces* E, F*, for* $\ell \in \mathbb{N}$*, the* ℓ-th entropy number *of* S *is defined as*

$$e_\ell(S) \triangleq \inf\left\{\epsilon > 0 : \exists s_1, \ldots, s_{2^{\ell-1}} \in S(\mathcal{B}_E) \text{ such that } S(\mathcal{B}_E) \subset \bigcup_{i=1}^{2^{\ell-1}} \mathcal{B}_F(s_i, \epsilon)\right\}.$$

The following lemma shows the relation between covering numbers and entropy numbers.

Lemma 3.B.3 (Relation between covering number and entropy number)**.** *Suppose a kernel $\boldsymbol{k}$ is jointly continuous and $A \subset \mathbb{R}^d$ is bounded. Then for any $\epsilon > 0$,*

$$e_{\lceil \log_2 \mathcal{N}_{\boldsymbol{k}}(A,\epsilon)\rceil+1}(\mathcal{H}_{\boldsymbol{k}|_A} \hookrightarrow \ell_\infty(A)) \leq \epsilon.$$

Proof of Lem. 3.B.3. First, the assumption implies $\boldsymbol{k}|_A$ is a bounded kernel, so by Steinwart and Christmann [SC08, Lemma 4.23], the inclusion $\mathcal{H}_{\boldsymbol{k}|_A} \hookrightarrow \ell_\infty(A)$ is continuous. By the definition of $\mathcal{N}_{\boldsymbol{k}|_A}(A,\epsilon)$, by adding arbitrary elements into the cover if necessary, there exists a $(\boldsymbol{k}|_A, A, \epsilon)$ cover of $\mathcal{B}_{\boldsymbol{k}|_A}$ of cardinality $2^{\lceil \log_2(\mathcal{N}_{\boldsymbol{k}|_A}(A,\epsilon))\rceil} \geq \mathcal{N}_{\boldsymbol{k}|_A}(A,\epsilon)$. Hence

$$e_{\lceil \log_2 \mathcal{N}_{\boldsymbol{k}|_A}(A,\epsilon)\rceil+1}(\mathcal{H}_{\boldsymbol{k}|_A} \hookrightarrow \ell_\infty(A)) \leq \epsilon.$$

The claim follows since $\mathcal{N}_{\boldsymbol{k}|_A}(A,\epsilon) = \mathcal{N}_{\boldsymbol{k}}(A,\epsilon)$ by Lem. 3.B.2. □

Proposition 3.B.1 (ℓ_∞-entropy number bound for PolyGrowth or LogGrowth $\boldsymbol{k}$)**.** *Suppose a kernel $\boldsymbol{k}$ satisfies Assum. (α, β)-kernel. Let $\mathfrak{C}_d > 0$ denote the constant that appears in the Assum. (α, β)-kernel. Define*

$$L_{\boldsymbol{k}}(r) \triangleq \tfrac{\mathfrak{C}_d}{\log 2} r^\beta. \tag{3.8}$$

Then for any $r > 0$ and $\ell \in \mathbb{N}$ that satisfies $\ell > L_{\boldsymbol{k}}(r+1) + 1$, we have

$$e_\ell(\mathcal{H}_{\boldsymbol{k}|_{\mathcal{B}_2(r)}} \hookrightarrow \ell_\infty(\mathcal{B}_2(r))) \leq \begin{cases} \left(\frac{L_{\boldsymbol{k}}(r+1)}{\ell-1}\right)^{\frac{1}{\alpha}} & \text{if } \boldsymbol{k} \text{ is PolyGrowth}(\alpha,\beta), \text{ and} \\ \exp\left(1 - \left(\frac{\ell-1}{L_{\boldsymbol{k}}(r+1)}\right)^{\frac{1}{\alpha}}\right) & \text{if } \boldsymbol{k} \text{ is LogGrowth}(\alpha,\beta). \end{cases}$$

Proof of Prop. 3.B.1. By Lem. 3.B.3 and the fact that e_ℓ is monotonically decreasing in ℓ by definition, if $\ell \geq \log_2 \mathcal{N}_{\boldsymbol{k}}(\mathcal{B}_2(r),\epsilon) + 1$ for some $\epsilon > 0$, then

$$e_\ell(\mathcal{H}_{\boldsymbol{k}|_{\mathcal{B}_2(r)}} \hookrightarrow \ell_\infty(\mathcal{B}_2(r))) \leq e_{\lceil \log_2 \mathcal{N}_{\boldsymbol{k}}(\mathcal{B}_2(r),\epsilon)\rceil+1}(\mathcal{H}_{\boldsymbol{k}|_{\mathcal{B}_2(r)}} \hookrightarrow \ell_\infty(\mathcal{B}_2(r))) \leq \epsilon. \tag{3.9}$$

For the PolyGrowth case, by its definition, the condition $\ell \geq \log_2 \mathcal{N}_{\boldsymbol{k}}(\mathcal{B}_2(r),\epsilon) + 1$ is met if $\epsilon \in (0,1)$ and

$$\ell \geq \tfrac{\mathfrak{C}_d}{\log 2}(1/\epsilon)^\alpha (r+1)^\beta + 1 \iff \epsilon \leq \left(\tfrac{L_{\boldsymbol{k}}(r+1)}{\ell-1}\right)^{\frac{1}{\alpha}}.$$

Hence (3.9) holds with $\epsilon = \left(\frac{L_{\boldsymbol{k}}(r+1)}{\ell-1}\right)^{\frac{1}{\alpha}}$, as long as $\epsilon \in (0,1)$, so ℓ needs to satisfy

$$1 > \left(\tfrac{L_{\boldsymbol{k}}(r+1)}{\ell-1}\right)^{\frac{1}{\alpha}} \iff \ell > L_{\boldsymbol{k}}(r+1) + 1.$$

Similarly, for the LogGrowth case, the condition $\ell \geq \log_2 \mathcal{N}_{\boldsymbol{k}}(\mathcal{B}_2(r), \epsilon) + 1$ is met if $\epsilon \in (0, 1)$ and

$$\ell \geq \tfrac{\mathfrak{C}_d}{\log 2}(\log(1/\epsilon) + 1)^\alpha (r+1)^\beta + 1 \iff \epsilon \leq \exp(1 - \left(\tfrac{\ell-1}{L_{\boldsymbol{k}}(r+1)}\right)^{\frac{1}{\alpha}}).$$

Hence (3.9) holds with $\epsilon = \exp(1 - \left(\tfrac{\ell-1}{L_{\boldsymbol{k}}(r+1)}\right)^{\frac{1}{\alpha}})$, as long as $\epsilon \in (0, 1)$, so ℓ needs to satisfy

$$1 > \exp(1 - \left(\tfrac{\ell-1}{L_{\boldsymbol{k}}(r+1)}\right)^{\frac{1}{\alpha}}) \iff \ell > L_{\boldsymbol{k}}(r+1) + 1.$$

□

Next, we show that we can transfer bounds on entropy numbers to obtain bounds for the eigenvalues of kernel matrices, which will become handy when we develop sub-quadratic-time algorithms in Sec. 3.3. We rely on the following lemma, which summarizes the relevant facts from Steinwart and Christmann [SC08, Appendix A].

Lemma 3.B.4 (Eigenvalue is bounded by entropy number)**.** *Let $\boldsymbol{k}$ be a jointly continuous kernel and $\mathbb{P}$ be a distribution such that $\mathbb{E}_{x\sim\mathbb{P}}[\boldsymbol{k}(x,x)] < \infty$, and recall that $\lambda_\ell(\cdot)$ denotes the ℓ-th largest eigenvalue of a linear operator. Then, for all $\ell \in \mathbb{N}$,*

$$\lambda_\ell(T_{\boldsymbol{k},\mathbb{P}}) \leq 4e_\ell^2(\mathcal{H}_{\boldsymbol{k}} \hookrightarrow \mathcal{L}^2(\mathbb{P})).$$

Proof of Lem. 3.B.4. For any bounded linear operator $S : \mathcal{H}_1 \to \mathcal{H}_2$ between Hilbert spaces $\mathcal{H}_1$ and $\mathcal{H}_2$, we have $a_\ell(S) \leq 2e_\ell(S)$, where a_ℓ is the ℓ-th approximation number defined in Steinwart and Christmann [SC08, (A.29)]. Recall the operator $S^*_{\boldsymbol{k},\mathbb{P}} = \mathcal{H}_{\boldsymbol{k}} \hookrightarrow \mathcal{L}^2(\mathbb{P})$ from (3.5), which is compact (in particular bounded) by Lem. 3.B.1(a). Thus

$$s_\ell(S^*_{\boldsymbol{k},\mathbb{P}}) = a_\ell(S^*_{\boldsymbol{k},\mathbb{P}}) \leq 2e_\ell(S^*_{\boldsymbol{k},\mathbb{P}}),$$

where the first equality follows from the paragraph below Steinwart and Christmann [SC08, (A.29)]) and s_ℓ is ℓ-th singular number of an operator [SC08, (A.25)]. Then using the identities mentioned under Steinwart and Christmann [SC08, (A.25)] and Steinwart and Christmann [SC08, (A.27)] and that all operators involved are compact by Lem. 3.B.1(b), we have

$$\lambda_\ell(T_{\boldsymbol{k},\mathbb{P}}) = \lambda_\ell(S^*_{\boldsymbol{k},\mathbb{P}} S_{\boldsymbol{k},\mathbb{P}}) = s_\ell(S^*_{\boldsymbol{k},\mathbb{P}} S_{\boldsymbol{k},\mathbb{P}}) = s_\ell^2(S^*_{\boldsymbol{k},\mathbb{P}}) \leq 4e_\ell^2(S^*_{\boldsymbol{k},\mathbb{P}}).$$

□

The previous lemma allows us to bound eigenvalues of kernel matrices by ℓ_∞-entropy numbers.

Proposition 3.B.2 (Eigenvalue of kernel matrix is bounded by ℓ_∞-entropy number)**.** *Let $\boldsymbol{k}$ be a jointly continuous kernel. Define $K \triangleq \boldsymbol{k}(\mathcal{S}_n, \mathcal{S}_n)$ for the sequence of points $\mathcal{S}_n = (x_1, \dots, x_n) \subset \mathbb{R}^d$. For any $w \in \Delta_{n-1}$, recall the notation $\mathbb{S}_n^w = \sum_{i\in[n]} w_i \delta_{x_i}$, $K^w = \operatorname{diag}(\sqrt{w}) K \operatorname{diag}(\sqrt{w})$, and $R_n = 1 + \sup_{i\in[n]} \|x_i\|_2$. Then for all $\ell \in \mathbb{N}$,*

$$\lambda_\ell(K^w) \overset{(i)}{=} \lambda_\ell(T_{\boldsymbol{k},\mathbb{S}_n^w}) \overset{(ii)}{\leq} 4e_\ell^2(\mathcal{H}_{\boldsymbol{k}|_{\mathcal{B}_2(R_n-1)}} \hookrightarrow \ell_\infty(\mathcal{B}_2(R_n-1))). \tag{3.10}$$

Proof of Prop. 3.B.2. Without loss of generality, we assume $w_i > 0$ for all $i \in [n]$, since otherwise, we can consider a smaller set of points by removing the ones with zero weights.

Proof of equality (i) from display (3.10) Note that $\mathcal{L}^2(\mathbb{S}_n^w)$ is isometric to $\mathbb{R}^n$. Let $K \triangleq \boldsymbol{k}(\mathcal{S}_n, \mathcal{S}_n)$ denote the kernel matrix. The action of $T_{\boldsymbol{k},\mathbb{S}_n^w}$ is given by, for $i \in [n]$,

$$T_{\boldsymbol{k},\mathbb{S}_n^w} f(x_i) = \textstyle\sum_{j\in[n]} w_j \boldsymbol{k}(x_i, x_j) f(x_j),$$

so in matrix form, $T_{\boldsymbol{k},\mathbb{S}_n^w} f = K \operatorname{diag}(w) f$, and hence $T_{\boldsymbol{k},\mathbb{S}_n^w} = K \operatorname{diag}(w)$. If λ_ℓ is an eigenvalue of $K \operatorname{diag}(w)$ with eigenvector v_ℓ, then

$$\begin{aligned} K \operatorname{diag}(w) v_\ell = \lambda_\ell v_\ell &\iff \operatorname{diag}(\sqrt{w}) K \operatorname{diag}(w) v_\ell = \lambda_\ell \operatorname{diag}(\sqrt{w}) v_\ell \\ &\iff \operatorname{diag}(\sqrt{w}) K \operatorname{diag}(\sqrt{w}) (\operatorname{diag}(\sqrt{w}) v_\ell) = \lambda_\ell \operatorname{diag}(\sqrt{w}) v_\ell, \end{aligned}$$

where we used $w_i > 0$ for all $i \in [n]$. Hence the eigenspectrum of $T_{\boldsymbol{k},\mathbb{S}_n^w}$ agrees with that of $\operatorname{diag}(\sqrt{w}) K \operatorname{diag}(\sqrt{w})$.

Proof of bound (ii) from display (3.10) By Lem. 3.B.4, we have $\lambda_\ell(T_{\boldsymbol{k},\mathbb{S}_n^w}) \leq 4e_\ell^2(\mathcal{H}_{\boldsymbol{k}|_{\mathcal{B}_2(R_n-1)}} \hookrightarrow \mathcal{L}^2(\mathbb{S}_n^w))$. Finally, using Def. 3.B.1, we have $e_\ell(\mathcal{H}_{\boldsymbol{k}|_{\mathcal{B}_2(R_n-1)}} \hookrightarrow \mathcal{L}^2(\mathbb{S}_n^w)) \leq e_\ell(\mathcal{H}_{\boldsymbol{k}|_{\mathcal{B}_2(R_n-1)}} \hookrightarrow \ell_\infty(\mathcal{B}_2(R_n-1)))$ because $\mathbb{S}_n^w$ is supported in $\mathcal{B}_2(R_n-1)$ and the fact that $\|\cdot\|_{\mathcal{L}^2(\mathbb{P})} \leq \|\cdot\|_\infty$ for any $\mathbb{P}$. □

Combining the tools developed so far, we have the following corollary for bounding the eigenvalues of PolyGrowth and LogGrowth kernel matrices.

Corollary 3.B.1 (Eigenvalue bound for PolyGrowth or LogGrowth kernel matrix)**.** *Suppose a kernel $\boldsymbol{k}$ satisfies Assum. ($\boldsymbol{\alpha}, \boldsymbol{\beta}$)-kernel. Let $\mathcal{S}_n = (x_1, \dots, x_n) \subset \mathbb{R}^d$ be a sequence of points. For any $w \in \Delta_{n-1}$, using the notation $L_{\boldsymbol{k}}$ from (3.8), for any $\ell > L_{\boldsymbol{k}}(R_n) + 1$, we have*

$$\lambda_\ell(K^w) \leq \begin{cases} 4\left(\frac{L_{\boldsymbol{k}}(R_n)}{\ell-1}\right)^{\frac{2}{\alpha}} & \textsc{PolyGrowth}(\alpha, \beta) \quad \textit{and} \\ 4\exp(2 - 2\left(\frac{\ell-1}{L_{\boldsymbol{k}}(R_n)}\right)^{\frac{1}{\alpha}}) & \textsc{LogGrowth}(\alpha, \beta). \end{cases}$$

Proof of Cor. 3.B.1. The claim follows by applying Prop. 3.B.2 and Prop. 3.B.1. □

♢ 3.B.2 Spectral decay of Stein kernels

The goal of this section is to show that a Stein kernel $\boldsymbol{k}_p$ satisfies Assum. ($\boldsymbol{\alpha}, \boldsymbol{\beta}$)-kernel provided that the base kernel is sufficiently smooth and to derive the parameters α, β for PolyGrowth and LogGrowth cases.

For a Stein kernel $\boldsymbol{k}_p$ with preconditioning matrix M, we define

$$\mathfrak{S}_p(r) \triangleq \max\left(1, \sup_{\|x\|_2 \le r} \left\|M^{1/2}\nabla \log p(x)\right\|_2\right). \tag{3.11}$$

We start by noting a useful alternative expression for a Stein kernel where we only need access to the density via the score $\nabla \log p$.

Proposition 3.B.3 (Alternative expression for Stein kernel)**.** *The Stein kernel $\boldsymbol{k}_p$ has the following alternative form:*

$$\begin{aligned}\boldsymbol{k}_p(x,y) = \langle \nabla \log p(x), M\nabla \log p(y)\rangle \boldsymbol{k}(x,y) + \langle \nabla \log p(x), M\nabla_y \boldsymbol{k}(x,y)\rangle + \\ \langle \nabla \log p(y), M\nabla_x \boldsymbol{k}(x,y)\rangle + \operatorname{tr}(M\nabla_x\nabla_y \boldsymbol{k}(x,y)),\end{aligned} \tag{3.12}$$

where $\nabla_x\nabla_y \boldsymbol{k}(x,y)$ denotes the $d \times d$ matrix $(\partial_{x_i}\partial_{y_j}\boldsymbol{k}(x,y))_{i,j\in[d]}$.

Proof of Prop. 3.B.3. We compute

$$\begin{aligned}(\nabla_x \cdot (p(x)M\boldsymbol{k}(x,y)p(y)))_j &= \textstyle\sum_{i\in[d]} M_{ij}\left(\partial_{x_i}p(x)\boldsymbol{k}(x,y)p(y) + p(x)\partial_{x_i}\boldsymbol{k}(x,y)p(y)\right).\\ \nabla_y \cdot \nabla_x \cdot (p(x)M\boldsymbol{k}(x,y)p(y)) &= \textstyle\sum_{i,j\in[d]} M_{ij}\left(\partial_{x_i}p(x)\partial_{y_j}p(y)\boldsymbol{k}(x,y) + \partial_{x_i}p(x)\partial_{y_j}\boldsymbol{k}(x,y)p(y)\right)\\ &\quad + \textstyle\sum_{i,j\in[d]} M_{ij}\left(p(x)\partial_{y_j}p(y)\partial_{x_i}\boldsymbol{k}(x,y) + p(x)\partial_{x_i}\partial_{y_j}\boldsymbol{k}(x,y)p(y)\right).\end{aligned}$$

The four terms in the last equation correspond to the four terms in (3.12). □

In what follows, we will make use of a matrix-valued kernel $\boldsymbol{K} : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^{d\times d}$ which generates an RKHS $\mathcal{H}_{\boldsymbol{K}}$ of vector-valued functions. See Carmeli, De Vito, and Toigo [CDT06] for an introduction to vector-valued RKHS theory.

Our next goal is to build a Hilbert-space isometry between the direct sum Hilbert space $\mathcal{H}_{\boldsymbol{k}}^{\oplus d}$ and $\mathcal{H}_{\boldsymbol{k}_p}$ to represent functions in $\mathcal{H}_{\boldsymbol{k}_p}$ using functions from $\mathcal{H}_{\boldsymbol{k}}$.

Lemma 3.B.5 (Preconditioned matrix-valued RKHS from a scalar kernel)**.** *Let $\boldsymbol{k} : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ be kernel and $\mathcal{H}_{\boldsymbol{k}}$ be the corresponding RKHS. Let $M \in \mathbb{R}^{d\times d}$ be an SPSD matrix. Consider the map $\iota : \mathcal{H}_{\boldsymbol{k}}^{\oplus d} \to \mathfrak{F}(\mathbb{R}^d, \mathbb{R}^d)$ defined by $(f_1, \dots, f_d) \mapsto [x \mapsto M^{1/2}(f_1(x), \dots, f_d(x))]$, where $\mathcal{H}_{\boldsymbol{k}}^{\oplus d}$ is the direct-sum Hilbert space of d copies of $\mathcal{H}_{\boldsymbol{k}}$ Then ι is a Hilbert-space isometry onto a vector-valued RKHS $\mathcal{H}_{\boldsymbol{K}}$ with matrix-valued reproducing kernel given by $\boldsymbol{K}(x,y) = \boldsymbol{k}(x,y)M$.*

Proof of Lem. 3.B.5. Define $\gamma : \mathbb{R}^d \to \mathfrak{F}(\mathbb{R}^d, {\mathcal{H}_{\boldsymbol{k}}}^{\oplus d})$ via

$$\gamma(x)(\alpha) \triangleq \boldsymbol{k}(x, \cdot) M^{1/2} \alpha.$$

We have

$$\|\gamma(x)(\alpha)\|_{{\mathcal{H}_{\boldsymbol{k}}}^{\oplus d}} \leq \|\boldsymbol{k}(x, \cdot)\|_{\boldsymbol{k}} \left\|M^{1/2}\right\|_2 \|\alpha\|_2,$$

so $\gamma(x)$ is bounded. Since $\gamma(x)$ is also linear, we have $\gamma(x) \in \mathfrak{B}(\mathbb{R}^d, {\mathcal{H}_{\boldsymbol{k}}}^{\oplus d})$. Let $\gamma(x)^* : {\mathcal{H}_{\boldsymbol{k}}}^{\oplus d} \to \mathbb{R}^d$ denote the Hilbert-space adjoint of $\gamma(x)$. Then for any $(f_1, \ldots, f_d) \in {\mathcal{H}_{\boldsymbol{k}}}^{\oplus d}$, $\alpha \in \mathbb{R}^d$, we have

$$\begin{aligned}
\langle \gamma(x)^*(f_1, \ldots, f_d), \alpha \rangle &= \langle (f_1, \ldots, f_d), \gamma(x)(\alpha) \rangle_{{\mathcal{H}_{\boldsymbol{k}}}^{\oplus d}} \\
&= \langle (f_1, \ldots, f_d), \boldsymbol{k}(x, \cdot) M^{1/2} \alpha \rangle_{{\mathcal{H}_{\boldsymbol{k}}}^{\oplus d}} \\
&= \langle (f_1(x), \ldots, f_d(x)), M^{1/2} \alpha \rangle \\
&= \langle M^{1/2}(f_1(x), \ldots, f_d(x)), \alpha \rangle.
\end{aligned}$$

Hence $\gamma(x)^*(f_1, \ldots, f_d) = M^{1/2}(f_1(x), \ldots, f_d(x))$, so $\iota(f_1, \ldots, f_d)(x) = \gamma(x)^*(f_1, \ldots, f_d)$. By Carmeli, De Vito, and Toigo [CDT06, Proposition 2.4], we see that ι is a partial isometry from ${\mathcal{H}_{\boldsymbol{k}}}^{\oplus d}$ onto a vector-valued RKHS space withv reproducing kernel $\boldsymbol{K}(x, y) = \gamma(x)^* \gamma(y) : \mathbb{R}^d \to \mathbb{R}^d$. For $\alpha \in \mathbb{R}^d$, previous calculation implies

$$\gamma(x)^* \gamma(y)(\alpha) = \gamma(x)^*(\boldsymbol{k}(y, \cdot) M^{1/2} \alpha) = M^{1/2} \boldsymbol{k}(y, x) M^{1/2} \alpha = \boldsymbol{k}(x, y) M.$$

□

Lemma 3.B.6 (Stein operator is an isometry)**.** *Consider a Stein kernel $\boldsymbol{k}_p$ with base kernel $\boldsymbol{k}$ and preconditioning matrix M. Then, the Stein operator $\mathcal{S}_p$ defined by $\mathcal{S}_p(v) \triangleq \frac{1}{p} \nabla \cdot (pv)$ is an isometry from $\mathcal{H}_{\boldsymbol{K}}$ with $\boldsymbol{K} \triangleq \boldsymbol{k} M$ to $\mathcal{H}_{\boldsymbol{k}_p}$.*

Proof. This follows from Barp et al. [Bar+22, Theorem 2.6] applied to $\boldsymbol{K}$. □

The previous two lemmas show that $\mathcal{S}_p \circ \iota$ is a Hilbert space isometry from ${\mathcal{H}_{\boldsymbol{k}}}^{\oplus d}$ to $\mathcal{H}_{\boldsymbol{k}_p}$. Note that $\mathcal{S}_p(v) = \langle \nabla \log p, h \rangle + \nabla \cdot h$. Hence, we immediately have

$$\mathcal{H}_{\boldsymbol{k}_p} = \left\{ \langle \nabla \log p, M^{1/2} f \rangle + \nabla \cdot (M^{1/2} f) : f = (f_1, \ldots, f_d) \in {\mathcal{H}_{\boldsymbol{k}}}^{\oplus d} \right\}. \tag{3.13}$$

We next build a divergence RKHS which is one of the summands used to form $\mathcal{H}_{\boldsymbol{k}_p}$.

Lemma 3.B.7 (Divergence RKHS)**.** *Let $\boldsymbol{k} : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ be a continuously differentiable kernel. Let M be an SPSD matrix. Define $\nabla^{\otimes 2} \cdot (M\boldsymbol{k}) : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ via*

$$\nabla^{\otimes 2} \cdot (M\boldsymbol{k})(x, y) \triangleq \nabla_y \cdot \nabla_x \cdot (M\boldsymbol{k}(x, y)) = \operatorname{tr}(M \nabla_x \nabla_y \boldsymbol{k}(x, y)). \tag{3.14}$$

Then $\nabla^{\otimes 2} \cdot (M\boldsymbol{k})$ *is a kernel, and its RKHS* $\mathcal{H}_{\nabla^{\otimes 2} \cdot (M\boldsymbol{k})}$ *has the following explicit form*

$$\mathcal{H}_{\nabla^{\otimes 2} \cdot (M\boldsymbol{k})} = \nabla \cdot \mathcal{H}_{\boldsymbol{K}} = \left\{ \nabla \cdot (M^{1/2} f) : f = (f_1, \ldots, f_d) \in {\mathcal{H}_{\boldsymbol{k}}}^{\oplus d} \right\}, \tag{3.15}$$

where $\boldsymbol{K} = M\boldsymbol{k}$*. Moreover,* $\nabla \cdot : \mathcal{H}_{\boldsymbol{K}} \to \mathcal{H}_{\nabla^{\otimes 2} \cdot (M\boldsymbol{k})}$ *is an isometry.*

Proof of Lem. 3.B.7. First of all, by Steinwart and Christmann [SC08, Corollary 4.36], every $f \in \mathcal{H}_{\boldsymbol{k}}$ is continuously differentiable, so $\partial_{x_i} f$ exists. By Lem. 3.B.5, $\nabla \cdot \mathcal{H}_{\boldsymbol{K}}$ is well-defined and the right equality in (3.15) holds.

Define $\gamma : \mathbb{R}^d \to \mathfrak{F}(\mathbb{R}, \mathcal{H}_{\boldsymbol{K}})$ via

$$\gamma(x)(t) \triangleq t \textstyle\sum_{i=1}^d \partial_{x_i} \boldsymbol{K}(x, \cdot) e_i,$$

where $e_i \in \mathbb{R}^d$ is the ith standard basis vector in $\mathbb{R}^d$; by Barp et al. [Bar+22, Lemma C.8] we have $\partial_{x_i} \boldsymbol{K}(x, \cdot) e_i \in \mathcal{H}_{\boldsymbol{K}}$. Note that

$$\|\gamma(x)(t)\|_{\boldsymbol{K}} = |t| \left\| \textstyle\sum_{i=1}^d \partial_{x_i} \boldsymbol{K}(x, \cdot) e_i \right\|_{\boldsymbol{K}},$$

so $\gamma(x) \in \mathfrak{B}(\mathbb{R}, \mathcal{H}_{\boldsymbol{K}})$. The adjoint $\gamma(x)^* \in \mathfrak{B}(\mathcal{H}_{\boldsymbol{K}}, \mathbb{R})$ must satisfy, for any $h \in \mathcal{H}_{\boldsymbol{K}}$,

$$t\gamma(x)^* h = \langle h, \gamma(x)(t) \rangle_{\boldsymbol{K}} = \left\langle h, t \textstyle\sum_{i=1}^d \partial_{x_i} \boldsymbol{K}(x, \cdot) e_i \right\rangle_{\boldsymbol{K}} = t \nabla \cdot h,$$

where we used the fact [Bar+22, Lemma C.8] that, for $c \in \mathbb{R}^d$, $h \in \mathcal{H}_{\boldsymbol{K}}$, $\langle \partial_{x_i} \boldsymbol{K}(x, \cdot) c, h \rangle = c^\top \partial_{x_i} h(x)$. So we find $\gamma(x)^*(h) = \nabla \cdot h(x)$. By Carmeli, De Vito, and Toigo [CDT06, Proposition 2.4], the map $A : \mathcal{H}_{\boldsymbol{K}} \to \mathfrak{F}(\mathbb{R}^d, \mathbb{R})$ defined by $A(h)(x) = \gamma^*(x)(h) = \nabla \cdot h(x)$, i.e., $A = \nabla \cdot$, is a partial isometry from $\mathcal{H}_{\boldsymbol{K}}$ to an RKHS $\mathcal{H}_{\nabla \cdot \boldsymbol{K}}$ with reproducing kernel

$$\gamma(x)^* \gamma(y) = \nabla \cdot \left(\textstyle\sum_{i=1}^d \partial_{x_i} \boldsymbol{K}(x, \cdot) e_i \right)(y) = \nabla_y \cdot \nabla_x \cdot \boldsymbol{K}(x, y) = \nabla^{\otimes 2} \cdot (M\boldsymbol{k})(x, y).$$

□

The following lemma shows that we can project a covering of $\mathcal{B}_{\boldsymbol{k}}$ consisting of arbitrary functions to a covering using functions only in $\mathcal{B}_{\boldsymbol{k}}$ while inflating the covering radius by at most 2.

Lemma 3.B.8 (Projection of coverings into RKHS balls)**.** *Let* $\boldsymbol{k}$ *be a kernel,* $A \subset \mathbb{R}^d$ *be a set, and* $\epsilon > 0$*. Let* $\mathcal{C}$ *be a set of functions such that for any* $f \in \mathcal{B}_{\boldsymbol{k}}$*, there exists* $g \in \mathcal{C}$ *such that* $\|f - g\|_{\infty, A} \leq \epsilon$*. Then*

$$\mathcal{N}_{\boldsymbol{k}}(A, 2\epsilon) \leq |\mathcal{C}|.$$

Proof. We will build a $(\boldsymbol{k}, A, 2\epsilon)$ covering $\mathcal{C}'$ as follows. For any $h \in \mathcal{C}$, if there exists $h' \in \mathcal{B}_{\boldsymbol{k}}$ with $\|h' - h\|_{\infty,A} \leq \epsilon$, then we include h' in $\mathcal{C}'$. By construction, $|\mathcal{C}'| \leq |\mathcal{C}|$. Then, for any $f \in \mathcal{B}_{\boldsymbol{k}}$, by assumption, there exists $g \in \mathcal{C}$ such that $\|f - g\|_{\infty,A} \leq \epsilon$. By construction, there exists $g' \in \mathcal{C}'$ such that $\|g' - g\|_{\infty,A} \leq \epsilon$. Thus

$$\|f - g'\|_{\infty,A} \leq \|f - g\|_{\infty,A} + \|g - g'\|_{\infty,A} \leq 2\epsilon.$$

Hence C' is a $(\boldsymbol{k}, A, 2\epsilon)$ covering. □

We are now ready to bound the covering numbers of $\boldsymbol{k}_p$ by those of $\boldsymbol{k}$ and $\nabla^{\otimes 2} \cdot (M\boldsymbol{k})$. Our key insight towards this end is that any element in $\mathcal{H}_{\boldsymbol{k}_p}$ can be decomposed as a sum of functions originated from $\mathcal{H}_{\boldsymbol{k}}$ and a function from the divergence RKHS $\mathcal{H}_{\nabla^{\otimes 2}\cdot(M\boldsymbol{k})}$.

Lemma 3.B.9 (Upper bounding covering number of Stein kernel with that of its base kernel)**.** *Let $\boldsymbol{k}_p$ be a Stein kernel with density p and preconditioning matrix M. For any $A \subset \mathbb{R}^d$, $\epsilon_1, \epsilon_2 > 0$,*

$$\mathcal{N}_{\boldsymbol{k}_p}(A, \epsilon) \leq \mathcal{N}_{\boldsymbol{k}}(A, \epsilon_1)^d \mathcal{N}_{\nabla^{\otimes 2}\cdot(M\boldsymbol{k})}(A, \epsilon_2),$$

for $\epsilon = 2(\sqrt{d}\epsilon_1 \sup_{x\in A} \left\|M^{1/2}\nabla \log p(x)\right\| + \epsilon_2)$.

Proof of Lem. 3.B.9. Let $\mathcal{C}_{\boldsymbol{k}}$ be a $(\boldsymbol{k}, A, \epsilon_1)$ covering and $\mathcal{C}_{\nabla^{\otimes 2}\cdot(M\boldsymbol{k})}$ be a $(\nabla^{\otimes 2}\cdot(M\boldsymbol{k}), A, \epsilon_2)$ covering. Define $b \triangleq M^{1/2}\nabla \log p$. Form

$$\mathcal{C} \triangleq \left\{\langle b, \tilde{f}\rangle + \tilde{g} : \tilde{f} = (\tilde{f}_1, \ldots, \tilde{f}_d) \in (\mathcal{C}_{\boldsymbol{k}})^d, \tilde{g} \in \mathcal{C}_{\nabla^{\otimes 2}\cdot(M\boldsymbol{k})}\right\} \subset \mathfrak{F}(\mathbb{R}^d, \mathbb{R}).$$

Then $|\mathcal{C}| \leq |\mathcal{C}_{\boldsymbol{k}}|^d \left|\mathcal{C}_{\nabla^{\otimes 2}\cdot(M\boldsymbol{k})}\right|$. Let $\boldsymbol{K} \triangleq \boldsymbol{k}M$. For any $h \in \mathcal{B}_{\boldsymbol{k}_p}$, by (3.13), we can find $f = (f_1, \ldots, f_d) \in {\mathcal{H}_{\boldsymbol{k}}}^{\oplus d}$ with $f_i \in \mathcal{H}_{\boldsymbol{k}}$ such that

$$h = \mathcal{S}_p \circ \iota(f) = \langle \nabla \log p, M^{1/2} f\rangle + \nabla \cdot (M^{1/2} f) = \langle b, f\rangle + \nabla \cdot (M^{1/2} f).$$

Since ι and $\mathcal{S}_p$ are isometries, we have $f \in \mathcal{B}_{{\mathcal{H}_{\boldsymbol{k}}}^{\oplus d}}$. Since, for each i,

$$\|f_i\|_{\boldsymbol{k}} \leq \sqrt{\textstyle\sum_{j=1}^d \|f_j\|_{\boldsymbol{k}}^2} = \|f\|_{{\mathcal{H}_{\boldsymbol{k}}}^{\oplus d}} \leq 1,$$

we have $f_i \in \mathcal{B}_{\boldsymbol{k}}$. By Lem. 3.B.7, $\nabla\cdot : \mathcal{H}_{\boldsymbol{K}} \to \mathcal{H}_{\nabla^{\otimes 2}\cdot(M\boldsymbol{k})}$ is also an isometry, so $\nabla \cdot (M^{1/2} f) \in \mathcal{B}_{\nabla^{\otimes 2}\cdot(M\boldsymbol{k})}$. Thus there exist $\tilde{f}_i \in \mathcal{C}_{\boldsymbol{k}}$ for each i and $\tilde{g} \in \mathcal{C}_{\nabla^{\otimes 2}\cdot(M\boldsymbol{k})}$ such that

$$\left\|f_i - \tilde{f}_i\right\|_{\infty,A} \leq \epsilon_1, \quad \left\|\nabla \cdot (M^{1/2} f) - \tilde{g}\right\|_{\infty,A} \leq \epsilon_2.$$

Let

$$\tilde{h}(x) \triangleq \langle b, \tilde{f}\rangle + \tilde{g} \in \mathcal{C}.$$

Then for $x \in A$,

$$\begin{aligned}\left|h(x) - \tilde{h}(x)\right| &= \left|\langle b(x), f(x) - \tilde{f}(x)\rangle + \nabla \cdot (M^{1/2} f(x)) - \tilde{g}(x)\right| \\ &\le \|b(x)\| \sqrt{\textstyle\sum_{i=1}^d (f_i(x) - \tilde{f}_i(x))^2} + \left|\nabla \cdot (M^{1/2} f(x)) - \tilde{g}(x)\right| \\ &\le \sqrt{d}\epsilon_1 \|b(x)\| + \epsilon_2.\end{aligned}$$

Hence

$$\left\|h - \tilde{h}\right\|_{\infty, A} \le \sqrt{d}\epsilon_1 \sup_{x\in A} \|b(x)\| + \epsilon_2 \triangleq \epsilon_3.$$

Note that $\mathcal{C}$ that we constructed is not necessarily contained in $\mathcal{B}_{\boldsymbol{k}_p}$. By Lem. 3.B.8, we can get a $(\boldsymbol{k}_p, A, 2\epsilon_3)$ covering and we are done. □

Corollary 3.B.2 (Log-covering number bound for Stein kernel)**.** *Let $\boldsymbol{k}_p$ be a Stein kernel and $A \subset \mathbb{R}^d$. For any $r > 0$, $\epsilon > 0$,*

$$\log \mathcal{N}_{\boldsymbol{k}_p}(A, \epsilon) \le d \log \mathcal{N}_{\boldsymbol{k}} \left(A, \tfrac{\epsilon}{4\sqrt{d}\mathfrak{S}_p(r)}\right) + \log \mathcal{N}_{\nabla^{\otimes 2}\cdot(M\boldsymbol{k})} \left(A, \tfrac{\epsilon}{4}\right),$$

where $\mathfrak{S}_p$ is defined in (3.11).

Proof. This is direct from Lem. 3.B.9 with $\epsilon_1 = \frac{\epsilon}{4\sqrt{d}\mathfrak{S}_p(r)}$, $\epsilon_2 = \frac{\epsilon}{4}$. □

Case of differentiable base kernel

Definition 3.B.2 (s-times continuously differentiable kernel)**.** *A kernel $\boldsymbol{k}$ is s-*times continuously differentiable *for $s \in \mathbb{N}$ if all partial derivatives $\partial^{\alpha,\alpha}\boldsymbol{k}$ exist and are continuous for all multi-indices $\alpha \in \mathbb{N}_0^d$ with $|\alpha| \le s$.*

Proposition 3.B.4 (Covering number bound for $\boldsymbol{k}_p$ with differentiable base kernel)**.** *Suppose $\boldsymbol{k}_p$ is a Stein kernel with an s-times continuously differentiable base kernel $\boldsymbol{k}$ for $s \ge 2$. Then there exist a constant $\mathfrak{C}_d > 0$ depending only on $(s, d, \boldsymbol{k}, M)$ such that for any $r > 0, \epsilon \in (0, 1)$,*

$$\log \mathcal{N}_{\boldsymbol{k}_p}(\mathcal{B}_2(r), \epsilon) \le \mathfrak{C}_d r^d \mathfrak{S}_p^{d/s}(r)(1/\epsilon)^{d/(s-1)}.$$

Proof of Prop. 3.B.4. Since $\boldsymbol{k}$ is s-times continuously differentiable, the divergence kernel $\nabla^{\otimes 2} \cdot (M\boldsymbol{k})$ is $(s-1)$-times continuously differentiable. By Dwivedi and Mackey [DM22b, Proposition 2(b)], there exists constants c_1, c_2 depending only on $(s, d, \boldsymbol{k}, M)$ such that, for any $r > 0$, $\epsilon_1, \epsilon_2 > 0$,

$$\begin{aligned}\log \mathcal{N}_{\boldsymbol{k}} \left(\mathcal{B}_2(r), \epsilon_1\right) &\le c_1 r^d (1/\epsilon)^{d/s}, \\ \log \mathcal{N}_{\nabla^{\otimes 2}\cdot(M\boldsymbol{k})} \left(\mathcal{B}_2(r), \epsilon_2\right) &\le c_2 r^d (1/\epsilon)^{d/(s-1)}.\end{aligned}$$

By Cor. 3.B.2 with $A = \mathcal{B}_2(r)$, we have, for any $r > 0$ and $\epsilon \in (0, 1)$,

$$\begin{aligned}\log \mathcal{N}_{\boldsymbol{k}_p}(\mathcal{B}_2(r), \epsilon) &\leq c_1 d r^d (4\sqrt{d}\mathfrak{S}_p(r))^{d/s} (1/\epsilon)^{d/\epsilon} + c_2 r^d (4/\epsilon)^{d/(s-1)} \\ &\leq \mathfrak{C}_d r^d \mathfrak{S}_p^{d/s}(r) (1/\epsilon)^{d/(s-1)}\end{aligned}$$

for some $\mathfrak{C}_d > 0$ depending only on $(s, d, \boldsymbol{k}, M)$. □

Case of radially analytic base kernel

For a symmetric positive definite $M \in \mathbb{R}^{d\times d}$, we define, for $x \in \mathbb{R}^d$,

$$\|x\|_M \triangleq \sqrt{x^\top M^{-1} x}.$$

Definition 3.B.3 (Radially analytic kernel)**.** *A kernel $\boldsymbol{k}$ is* radially analytic *if $\boldsymbol{k}$ satisfies $\boldsymbol{k}(x, y) = \kappa(\|x - y\|_M^2)$ for a symmetric positive definite matrix $M \in \mathbb{R}^{d\times d}$ and a function $\kappa : \mathbb{R}_{\geq 0} \to \mathbb{R}$ real-analytic everywhere with convergence radius $R_\kappa > 0$ such that there exists a constant $C_\kappa > 0$ for which*

$$\left|\tfrac{1}{j!}\kappa_+^{(j)}(0)\right| \leq C_\kappa (2/R_\kappa)^j, \text{ for all } j \in \mathbb{N}_0, \tag{3.16}$$

where $\kappa_+^{(j)}$ indicates the j-th right-sided derivative of κ.

Example 3.B.1 (Gaussian kernel)**.** *Consider the Gaussian kernel $\boldsymbol{k}(x, y) = \kappa(\|x - y\|_M^2)$ with $\kappa(t) = e^{-\frac{t}{2\sigma^2}}$ where $\sigma > 0$. Note the exponential function is real-analytic everywhere, and so is κ. Since $\kappa(t) = \sum_{j=0}^\infty \frac{(-t/2\sigma^2)^j}{j}$, we find $\frac{1}{j!}\kappa^{(j)}(0) = \frac{(-1)^j}{j(2\sigma^2)^j}$. Hence (3.16) holds with $C_\kappa = 1$ and $R_\kappa = 2\inf_{j\geq 0}(j(2\sigma^2)^j)^{1/j} = 4\sigma^2$.*

Example 3.B.2 (IMQ kernel)**.** *Consider the inverse multiquadric kernel $\boldsymbol{k}(x, y) = \kappa(\|x - y\|_M^2)$ with $\kappa(t) = (c^2 + t)^{-\beta}$ where $c, \beta > 0$. By Sun and Zhou [SZ08, Example 3], κ is real-analytic everywhere with $C_\kappa = c^{-2\beta}(2\beta + 1)^{\beta+1}$ and $R_\kappa = c^2$.*

A simple calculation yields the following lemma.

Proposition 3.B.5 (Expression for $\boldsymbol{k}_p$ with a radially analytic base kernel)**.** *Suppose a Stein kernel $\boldsymbol{k}_p$ has a symmetric positive definite preconditioning matrix and a base kernel $\boldsymbol{k}(x, y) = \kappa(\|x - y\|_M^2)$ where κ is twice-differentiable. Then*

$$\begin{aligned}\boldsymbol{k}_p(x, y) = \langle \nabla \log p(x), M \nabla \log p(y)\rangle \kappa(\|x - y\|_M^2) - \\ 2\kappa_+'(\|x - y\|_M^2)\langle x - y, \nabla \log p(x) - \nabla \log p(y)\rangle - \\ 4\kappa_+''(\|x - y\|_M^2)\|x - y\|_M^2 - 2d\kappa_+'(\|x - y\|_M^2).\end{aligned} \tag{3.17}$$

In particular,

$$\boldsymbol{k}_p(x, x) = \left\|M^{1/2}\nabla \log p(x)\right\|_2^2 \kappa(0) - 2d\kappa_+'(0).$$

Proof of Prop. 3.B.5. From $\boldsymbol{k}(x,y) = \kappa(\|x-y\|_M^2) = \kappa((x-y)^\top M^{-1}(x-y))$, we compute, using (3.14),

$$\begin{aligned}
\nabla_y \boldsymbol{k}(x,y) &= -2\kappa'_+(\|x-y\|_M^2)M^{-1}(x-y) \\
\nabla_x\nabla_y \boldsymbol{k}(x,y) &= -2\kappa'_+(\|x-y\|_M^2)M^{-1} - 4\kappa''_+(\|x-y\|_M^2)M^{-1}(x-y)((x-y)M^{-1})^\top \\
\nabla^{\otimes 2}\cdot(M\boldsymbol{k})(x,y) &= \operatorname{tr}(M\nabla_x\nabla_y \boldsymbol{k}(x,y)) = -4\kappa''_+(\|x-y\|_M^2)\,\|x-y\|_M^2 - 2d\kappa'_+(\|x-y\|_M^2).
\end{aligned} \tag{3.18}$$

The form (3.17) then follows from applying Prop. 3.B.3. □

We next show that the divergence kernel $\nabla^{\otimes 2}\cdot(M\boldsymbol{k})$ is radially analytic given that $\boldsymbol{k}$ is.

Lemma 3.B.10 (Convergence radius of divergence kernel)**.** *Let $\boldsymbol{k}$ be a radially analytic kernel with the corresponding real-analytic function κ, convergence radius R_κ with constant C_κ, and a symmetric positive definite matrix M. Then*

$$\nabla^{\otimes 2}\cdot(M\boldsymbol{k})(x,y) = \kappa_{\nabla^{\otimes 2}\cdot(M\boldsymbol{k})}(\|x-y\|_M^2),$$

where $\kappa_{\nabla^{\otimes 2}\cdot(M\boldsymbol{k})} : \mathbb{R}_{\geq 0} \to \mathbb{R}$ is the real-analytic function defined as

$$\kappa_{\nabla^{\otimes 2}\cdot(M\boldsymbol{k})}(t) \triangleq -4\kappa''_+(t)t - 2d\kappa'_+(t).$$

Moreover, $\kappa_{\nabla^{\otimes 2}\cdot(M\boldsymbol{k})}$ has convergence radius with constant

$$R_{\kappa_{\nabla^{\otimes 2}\cdot(M\boldsymbol{k})}} = \tfrac{R_\kappa}{4d+8}, \quad C_{\kappa_{\nabla^{\otimes 2}\cdot(M\boldsymbol{k})}} = 4dC_\kappa/R_\kappa.$$

Proof of Lem. 3.B.10. The first statement regarding the form of $\kappa_{\nabla^{\otimes 2}\cdot(M\boldsymbol{k})}$ directly follows from (3.18). Next, iterative differentiation yields, for $j \in \mathbb{N}_0$,

$$\kappa^{(j)}_{\nabla^{\otimes 2}\cdot(M\boldsymbol{k})}(t) = -(2d+4j)\kappa_+^{(j+1)}(t) - 4\kappa_+^{(j+2)}(t)t.$$

Thus

$$\begin{aligned}
\left|\tfrac{1}{j!}\kappa^{(j)}_{\nabla^{\otimes 2}\cdot(M\boldsymbol{k})}(0)\right| &= \tfrac{2d+4j}{j!}\kappa_+^{(j+1)}(0) \\
&= \tfrac{(2d+4j)(j+1)}{(j+1)!}\kappa_+^{(j+1)}(0) \\
&\leq (2d+4j)(j+1)C_\kappa(2/R_\kappa)^{j+1}.
\end{aligned} \tag{3.19}$$

For $j \geq 1$,

$$\begin{aligned}
\left|\tfrac{1}{j!}\kappa^{(j)}_{\nabla^{\otimes 2}\cdot(M\boldsymbol{k})}(0)\right| &\leq (2C_\kappa/R_\kappa)\left(((2d+4j)(j+1))^{1/j}2/R_\kappa\right)^j \\
&\leq (2C_\kappa/R_\kappa)\left((2(2d+4)\cdot 2/R_\kappa\right)^j.
\end{aligned}$$

where we used the fact that $((2d+4j)(j+1))^{1/j}$ is decreasing in j. For $j=0$, (3.19) is just $2dC_\kappa\cdot 2/R_\kappa$. Hence $\kappa_{\nabla^{\otimes 2}\cdot(M\boldsymbol{k})}$ is analytic with $C_{\kappa_{\nabla^{\otimes 2}\cdot(M\boldsymbol{k})}} = 4dC_\kappa/R_\kappa$ and $R_{\kappa_{\nabla^{\otimes 2}\cdot(M\boldsymbol{k})}} = \tfrac{R_\kappa}{4d+8}$. □

We will use the following lemma repeatedly.

Lemma 3.B.11 (Covering number of radially analytic kernel with M-metric)**.** *Let $\boldsymbol{k}_0$ be a radially analytic kernel with $\boldsymbol{k}_0(x,y) = \kappa(\|x-y\|_2^2)$. For any symmetric positive definite $M \in \mathbb{R}^{d\times d}$, consider the radially analytic kernel $\boldsymbol{k}(x,y) \triangleq \kappa(\|x-y\|_M^2)$. Then for any $A \subset \mathbb{R}^d$ and $\epsilon > 0$, we have*

$$\mathcal{N}_{\boldsymbol{k}}(M^{-1/2}(A), \epsilon) = \mathcal{N}_{\boldsymbol{k}_0}(A, \epsilon).$$

In particular, for any $r > 0$,

$$\mathcal{N}_{\boldsymbol{k}}(\mathcal{B}_2(r), \epsilon) \le \mathcal{N}_{\boldsymbol{k}_0}(\mathcal{B}_2(r\|M^{1/2}\|_2), \epsilon).$$

Proof. Note that $\boldsymbol{k}(x,y) = \boldsymbol{k}_0(M^{-1/2}x, M^{-1/2}y)$. By Paulsen and Raghupathi [PR16, Theorem 5.7], $\mathcal{H}_{\boldsymbol{k}} = \{f \circ M^{-1/2} : f \in \mathcal{H}_{\boldsymbol{k}_0}\}$, and moreover $\mathcal{B}_{\boldsymbol{k}} = \{f \circ M^{-1} : f \in \mathcal{B}_{\boldsymbol{k}_0}\}$. Let $\mathcal{C}_0$ be a $(\boldsymbol{k}_0, A, \epsilon)$ covering. Form $\mathcal{C} = \{h \circ M^{-1/2} : h \in \mathcal{C}_0\} \subset \mathcal{B}_{\boldsymbol{k}}$. For any element $f \circ M^{-1/2} \in \mathcal{B}_{\boldsymbol{k}}$ where $f \in \mathcal{B}_{\boldsymbol{k}_0}$, there exists $h \in \mathcal{C}_0$ such that $\|f-h\|_{\infty,A} \le \epsilon$. Thus

$$\left\|f \circ M^{-1/2} - h \circ M^{-1/2}\right\|_{\infty, M^{-1/2}(A)} = \|f-h\|_{\infty,A} \le \epsilon.$$

Thus $\mathcal{N}_{\boldsymbol{k}}(M^{-1/2}(A), \epsilon) \le \mathcal{N}_{\boldsymbol{k}_0}(A, \epsilon)$. By considering M^{-1} in place of M, we get our desired equality.

For the second statement, by letting $A = M^{1/2}\mathcal{B}_2(r)$, we have

$$\mathcal{N}_{\boldsymbol{k}}(\mathcal{B}_2(r), \epsilon) = \mathcal{N}_{\boldsymbol{k}_0}(M^{1/2}\mathcal{B}_2(r), \epsilon) \le \mathcal{N}_{\boldsymbol{k}_0}(\mathcal{B}_2(r\|M^{1/2}\|_2), \epsilon),$$

where we use the fact that $M^{1/2}\mathcal{B}_2(r) \subset \mathcal{B}_2(r\|M^{1/2}\|_2)$. □

In the next lemma, we rephrase the result from Sun and Zhou [SZ08, Theorem 2] for bounding the covering number of a radially analytic kernel.

Lemma 3.B.12 (Covering number bound for radially analytic kernel)**.** *Let $\boldsymbol{k}$ be a radially analytic kernel with $\boldsymbol{k}(x,y) = \kappa(\|x-y\|_2^2)$. Then, there exist a polynomial $P(r)$ of degree $2d$ and a constant C depending only on (κ, d) such that for any $r > 0$, $\epsilon \in (0, 1/2)$,*

$$\log \mathcal{N}_{\boldsymbol{k}}(\mathcal{B}_2(r), \epsilon) \le P(r)(\log(1/\epsilon) + C)^{d+1}.$$

Proof of Lem. 3.B.12. Let R_κ, C_κ denote the constants for κ as in (3.16). By and Sun and Zhou [SZ08, Theorem 2] with $R = 1$, $D = 2r$, and Lem. 3.B.2, for $\epsilon \in (0, 1/2)$, we have

$$\log \mathcal{N}_{\boldsymbol{k}}(\mathcal{B}_2(r), \epsilon) \le N_2(\mathcal{B}_2(r), r^\dagger/2)\left(4\log(1/\epsilon) + 2 + 4\log(16\sqrt{C_\kappa} + 1)\right)^{d+1},$$

where $r^\dagger = \min(\frac{\sqrt{R_\kappa}}{2d}, \sqrt{R_\kappa + (2r)^2} - 2r)$, and $N_2(\mathcal{B}_2(r), r^\dagger/2)$ is the covering number of $\mathcal{B}_2(r)$ as a subset of $\mathbb{R}^d$, which can be further bounded by [Wai19, (5.8)]

$$N_2(\mathcal{B}_2(r), r^\dagger/2) \le \left(1 + \tfrac{4r}{r^\dagger}\right)^d.$$

If $r^\dagger = \sqrt{R_\kappa + (2r)^2} - 2r$, then $\frac{r}{r^\dagger} = \frac{r}{\sqrt{R_\kappa+(2r)^2}-2r} = \frac{r(\sqrt{R_\kappa+(2r)^2}+2r)}{R_\kappa} \le \frac{r(\sqrt{R_\kappa}+4r)}{R_\kappa}$ which is a quadratic polynomial in r. Hence for a constant $C > 0$ and a polynomial $P(r)$ of degree $2d$ that depend only on (κ, d), we have the claim. □

Proposition 3.B.6 (Covering number bound for $\boldsymbol{k}_p$ with radially analytic base kernel)**.** *Suppose $\boldsymbol{k}_p$ is a Stein kernel with a preconditioning matrix M and a radially analytic base kernel $\boldsymbol{k}$ based on a real-analytic function κ. Then there exist a constant $C > 0$ and a polynomial $P(r)$ of degree $2d$ depending only on (κ, d, M) such that for any $r > 0, \epsilon \in (0, 1)$,*

$$\log \mathcal{N}_{\boldsymbol{k}_p}(\mathcal{B}_2(r), \epsilon) \le \left(\log \tfrac{\mathfrak{S}_p(r)}{\epsilon} + C\right)^{d+1} P(r). \tag{3.20}$$

Proof of Prop. 3.B.6. Recall $\boldsymbol{k}(x, y) = \kappa(\|x - y\|_M^2)$. Consider $\boldsymbol{k}_0(x, y) \triangleq \kappa(\|x - y\|_2^2)$. For $\epsilon_1 \in (0, 1/2)$, by Lem. 3.B.11, we have

$$\log \mathcal{N}_{\boldsymbol{k}}(\mathcal{B}_2(r/\|M^{1/2}\|_2), \epsilon_1) \le \log \mathcal{N}_{\boldsymbol{k}_0}(\mathcal{B}_2(r), \epsilon_1/2).$$

Thus by Lem. 3.B.12, there exists a polynomial $P_{\boldsymbol{k}}(r)$ of degree $2d$ and a constant $C_{\boldsymbol{k}}$ depending only on (κ, d, M) such that

$$\log \mathcal{N}_{\boldsymbol{k}}(\mathcal{B}_2(r), \epsilon_1) \le P_{\boldsymbol{k}}(r)(\log(1/\epsilon_1) + C_{\boldsymbol{k}})^{d+1}$$

Similarly, for $\epsilon_2 \in (0, 1/2)$, by Lem. 3.B.10 and Lem. 3.B.12, we have, for a constant $C_{\nabla^{\otimes 2}\cdot(M\boldsymbol{k})} > 0$ and a polynomial $P_{\nabla^{\otimes 2}\cdot(M\boldsymbol{k})}(r)$ of degree $2d$ that depend only on (κ, d, M),

$$\log \mathcal{N}_{\nabla^{\otimes 2}\cdot(M\boldsymbol{k})}(\mathcal{B}_2(r), \epsilon_2) \le P_{\nabla^{\otimes 2}\cdot(M\boldsymbol{k})}(r)(\log(1/\epsilon_2) + C_{\nabla^{\otimes 2}\cdot(M\boldsymbol{k})})^{d+1}.$$

For a given $\epsilon \in (0, 1)$, let $\epsilon_1 = \frac{\epsilon}{4\sqrt{d}\mathfrak{S}_p(r)}$ and $\epsilon_2 = \frac{\epsilon}{4}$. Then since $\mathfrak{S}_p \ge 1$, we have $\epsilon_1, \epsilon_2 \in (0, 1/2)$. By Cor. 3.B.2 with $A = \mathcal{B}_2(r)$, we obtain, for a constants $C > 0$ and a polynomial $P(r)$ of degree $2d$ that depend only on (κ, d, M),

$$\log \mathcal{N}_{\boldsymbol{k}_p}(\mathcal{B}_2(r), \epsilon) \le P(r)(\log(1/\epsilon) + \log \mathfrak{S}_p(r) + C)^{d+1}.$$

Hence (3.20) is shown. □

When $\log \mathfrak{S}_p(r)$ grows polynomially in r, we apply Prop. 3.B.6 to immediately obtain the following.

Corollary 3.B.3. *Under the assumption of Prop. 3.B.6, suppose* $\mathfrak{S}_p(r) = O(\mathrm{poly}(r))$. *Then for any* $\delta > 0$, *there exists* $\mathfrak{C}_d > 0$ *such that*

$$\log \mathcal{N}_{\boldsymbol{k}_p}(\mathcal{B}_2(r), \epsilon) \leq \mathfrak{C}_d \log(e/\epsilon)^{d+1}(r+1)^{2d+\delta}.$$

Proof of Cor. 3.B.3. This immediately follows from Prop. 3.B.6 by using $\delta > 0$ to absorb the $\log \mathfrak{S}_p(r) = O(r^\delta)$ term. □

Proof of Prop. 3.1: Stein kernel growth rates

This follows from Prop. 3.B.4 and Cor. 3.B.3, and by noticing that if $\sup_{\|x\|_2 \leq r} \|\nabla \log p(x)\|_2$ is bounded by a degree d_ℓ polynomial, then so is

$$\mathfrak{S}_p(r) = \sup_{\|x\|_2 \leq r} \left\|M^{1/2} \nabla \log p(x)\right\|_2 \leq \left\|M^{1/2}\right\|_2 \sup_{\|x\|_2 \leq r} \|\nabla \log p(x)\|_2 .$$

□

■ 3.C A Debiasing Benchmark

◇ 3.C.1 MMD of unbiased i.i.d. sample points

We start by showing that sequence of n points sampled i.i.d. from $\mathbb{P}$ achieves $\Theta(n^{-1})$ squared $\mathrm{MMD}_{\boldsymbol{k}_\mathbb{P}}$ to $\mathbb{P}$ in expectation.

Proposition 3.C.1 (MMD of unbiased i.i.d. sample points)**.** *Let* $\boldsymbol{k}_\mathbb{P}$ *be a kernel satisfying Assum. 3.1 with* $\mathfrak{p} \geq 1$. *Let* $\mathcal{S}_n = (x_i)_{i \in [n]}$ *be* n *i.i.d. samples from* $\mathbb{P}$. *Then*

$$\mathbb{E}[\mathrm{MMD}_{\boldsymbol{k}_\mathbb{P}}(\mathbb{S}_n, \mathbb{P})^2] = \tfrac{\mathbb{E}_{x \sim \mathbb{P}}[\boldsymbol{k}_\mathbb{P}(x,x)]}{n}.$$

Proof of Prop. 3.C.1. We compute

$$\mathbb{E}[\mathrm{MMD}_{\boldsymbol{k}_\mathbb{P}}(\mathbb{S}_n, \mathbb{P})^2] = \mathbb{E}[\textstyle\sum_{i,j \in [n]} \frac{1}{n^2} \boldsymbol{k}_\mathbb{P}(x_i, x_j)] = \frac{1}{n^2} \sum_{i,j \in [n]} \mathbb{E}[\boldsymbol{k}_\mathbb{P}(x_i, x_j)] = \frac{1}{n} \mathbb{E}[\boldsymbol{k}_\mathbb{P}(x_1, x_1)],$$

where we used the fact that $\boldsymbol{k}_\mathbb{P}$ is mean-zero with respect to $\mathbb{P}$ and the independence of x_i, x_j for $i \neq j$. □

◇ 3.C.2 Proof of Thm. 3.1: Debiasing via simplex reweighting

We make use of the self-normalized importance sampling weights

$$w_j^{\mathrm{SNIS}} = \tfrac{\mathrm{d}\mathbb{P}}{\mathrm{d}\mathbb{Q}}(x_j) / \textstyle\sum_{i \in [n]} \frac{\mathrm{d}\mathbb{P}}{\mathrm{d}\mathbb{Q}}(x_i)$$

for $j \in [n]$ in our proofs. Notice that $(w_1^{\mathrm{SNIS}}, \dots, w_n^{\mathrm{SNIS}})^\top \in \Delta_{n-1}$ and hence

$$\mathrm{MMD}_{\mathrm{OPT}} \le \mathrm{MMD}_{\boldsymbol{k}_{\mathbb{P}}}(w_i^{\mathrm{SNIS}}\delta_{x_i}, \mathbb{P}) = \frac{\|\sum_{i=1}^n \frac{\mathrm{d}\mathbb{P}}{\mathrm{d}\mathbb{Q}}(x_i)\boldsymbol{k}_{\mathbb{P}}(x_i,\cdot)\|_{\boldsymbol{k}_{\mathbb{P}}}}{\sum_{i=1}^n \frac{\mathrm{d}\mathbb{P}}{\mathrm{d}\mathbb{Q}}(x_i)} = \frac{\|\frac{1}{n}\sum_{i=1}^n \frac{\mathrm{d}\mathbb{P}}{\mathrm{d}\mathbb{Q}}(x_i)\boldsymbol{k}_{\mathbb{P}}(x_i,\cdot)\|_{\boldsymbol{k}_{\mathbb{P}}}}{\frac{1}{n}\sum_{i=1}^n \frac{\mathrm{d}\mathbb{P}}{\mathrm{d}\mathbb{Q}}(x_i)}.$$

Introduce the bounded in probability notation $X_n = O_p(g_n)$ to mean $\Pr(|X_n/g_n| > C_\epsilon) \le \epsilon$ for all $n \ge N_\epsilon$ for any $\epsilon > 0$. Then we claim that under the conditions assumed in Thm. 3.1,

$$\|\tfrac{1}{n}\textstyle\sum_{i=1}^n \frac{\mathrm{d}\mathbb{P}}{\mathrm{d}\mathbb{Q}}(x_i)\boldsymbol{k}_{\mathbb{P}}(x_i,\cdot)\|_{\boldsymbol{k}_{\mathbb{P}}} = O_p(n^{-\frac{1}{2}}) \quad \text{and} \quad \tfrac{1}{n}\textstyle\sum_{i=1}^n \frac{\mathrm{d}\mathbb{P}}{\mathrm{d}\mathbb{Q}}(x_i) \to 1 \text{ almost surely}, \tag{3.21}$$

so that by Slutsky's theorem [Wel+13, Ex. 1.4.7], we have $\mathrm{MMD}_{\mathrm{OPT}} = O_p(n^{-\frac{1}{2}})$ as desired. We prove the claims in (3.21) in two main steps: (a) first, we construct a weighted RKHS and then (b) establish a central limit theorem (CLT) that allows us to conclude both claims from (3.21)

Constructing a weighted and separable RKHS Define the kernel

$$\boldsymbol{k}_{\mathbb{Q}}(x,y) \triangleq \frac{\mathrm{d}\mathbb{P}}{\mathrm{d}\mathbb{Q}}(x)\boldsymbol{k}_{\mathbb{P}}(x,y)\frac{\mathrm{d}\mathbb{P}}{\mathrm{d}\mathbb{Q}}(y)$$

with Hilbert space $\mathcal{H}_{\boldsymbol{k}_{\mathbb{Q}}} = \frac{\mathrm{d}\mathbb{P}}{\mathrm{d}\mathbb{Q}}\mathcal{H}_{\boldsymbol{k}_{\mathbb{P}}}$ and the elements $\xi_i \triangleq \boldsymbol{k}_{\mathbb{Q}}(x_i,\cdot) = \frac{\mathrm{d}\mathbb{P}}{\mathrm{d}\mathbb{Q}}(x_i)\boldsymbol{k}_{\mathbb{P}}(x_i,\cdot)\frac{\mathrm{d}\mathbb{P}}{\mathrm{d}\mathbb{Q}}(\cdot) \in \mathcal{H}_{\boldsymbol{k}_{\mathbb{Q}}}$ for each $i \in \mathbb{N}$. By Paulsen and Raghupathi [PR16, Prop. 5.20], any element in $\mathcal{H}_{\boldsymbol{k}_{\mathbb{Q}}}$ is represented by $\frac{\mathrm{d}\mathbb{P}}{\mathrm{d}\mathbb{Q}}f$ for some $f \in \mathcal{H}_{\boldsymbol{k}_{\mathbb{P}}}$ and moreover, $f \mapsto \frac{\mathrm{d}\mathbb{P}}{\mathrm{d}\mathbb{Q}}f$ preserves inner product between the two RKHSs, i.e., $\langle f, g\rangle_{\boldsymbol{k}_{\mathbb{P}}} = \langle \frac{\mathrm{d}\mathbb{P}}{\mathrm{d}\mathbb{Q}}f, \frac{\mathrm{d}\mathbb{P}}{\mathrm{d}\mathbb{Q}}g\rangle_{\boldsymbol{k}_{\mathbb{Q}}}$ for $f, g \in \mathcal{H}_{\boldsymbol{k}_{\mathbb{P}}}$, which in turn implies $\|f\|_{\boldsymbol{k}_{\mathbb{P}}} = \|\frac{\mathrm{d}\mathbb{P}}{\mathrm{d}\mathbb{Q}}f\|_{\boldsymbol{k}_{\mathbb{Q}}}$. As a result, we also have that

$$\|\tfrac{1}{n}\textstyle\sum_{i=1}^n \frac{\mathrm{d}\mathbb{P}}{\mathrm{d}\mathbb{Q}}(x_i)\boldsymbol{k}_{\mathbb{P}}(x_i,\cdot)\|_{\boldsymbol{k}_{\mathbb{P}}} = \|\tfrac{1}{n}\textstyle\sum_{i=1}^n \frac{\mathrm{d}\mathbb{P}}{\mathrm{d}\mathbb{Q}}(x_i)\boldsymbol{k}_{\mathbb{P}}(x_i,\cdot)\frac{\mathrm{d}\mathbb{P}}{\mathrm{d}\mathbb{Q}}(\cdot)\|_{\boldsymbol{k}_{\mathbb{Q}}} = \|\tfrac{1}{n}\textstyle\sum_{i=1}^n \xi_i\|_{\boldsymbol{k}_{\mathbb{Q}}}. \tag{3.22}$$

Since $\mathcal{H}_{\boldsymbol{k}_{\mathbb{P}}}$ is separable, there exists a dense countable subset $(f_n)_{n\in\mathbb{N}} \subset \mathcal{H}_{\boldsymbol{k}_{\mathbb{P}}}$. For any $\frac{\mathrm{d}\mathbb{P}}{\mathrm{d}\mathbb{Q}}f \in \mathcal{H}_{\boldsymbol{k}_{\mathbb{Q}}}$, there exists $\{n_k\}_{k\in\mathbb{N}}$ such that $\lim_{k\to\infty}\|f_{n_k} - f\|_{\boldsymbol{k}_{\mathbb{P}}} = 0$. Since

$$\|\tfrac{\mathrm{d}\mathbb{P}}{\mathrm{d}\mathbb{Q}}f_{n_k} - \tfrac{\mathrm{d}\mathbb{P}}{\mathrm{d}\mathbb{Q}}f\|_{\boldsymbol{k}_{\mathbb{Q}}} = \|\tfrac{\mathrm{d}\mathbb{P}}{\mathrm{d}\mathbb{Q}}(f_{n_k} - f)\|_{\boldsymbol{k}_{\mathbb{Q}}} = \|f_{n_k} - f\|_{\boldsymbol{k}_{\mathbb{P}}}$$

due to inner-product preservation, we thus have

$$\lim_{k\to\infty}\|\tfrac{\mathrm{d}\mathbb{P}}{\mathrm{d}\mathbb{Q}}f_{n_k} - \tfrac{\mathrm{d}\mathbb{P}}{\mathrm{d}\mathbb{Q}}f\|_{\boldsymbol{k}_{\mathbb{Q}}} = \lim_{k\to\infty}\|f_{n_k} - f\|_{\boldsymbol{k}_{\mathbb{P}}}0,$$

so $(\frac{\mathrm{d}\mathbb{P}}{\mathrm{d}\mathbb{Q}}f_n)_{n\in\mathbb{N}}$ is dense in $\mathcal{H}_{\boldsymbol{k}_{\mathbb{Q}}}$, showing that $\mathcal{H}_{\boldsymbol{k}_{\mathbb{Q}}}$ is separable.

Harris recurrence of the chain $(x_i)_{i\in\mathbb{N}}$ Let μ_1 denote the distribution of x_1. Since $\mathcal{S}_\infty = (x_i)_{i=1}^\infty$ is a homogeneous ϕ-irreducible geometrically ergodic Markov chain with stationary distribution $\mathbb{Q}$, it is also positive [MT12, Ch. 10] by definition and aperiodic by Douc et al. [Dou+18, Lem. 9.3.9]. Moreover, since $\mathcal{S}_\infty$ is ϕ-irreducible, aperiodic, and geometrically ergodic in the sense of Gallegos-Herrada, Ledvinka, and Rosenthal [GLR23, Thm. 1] and μ_1 is absolutely continuous with respect to $\mathbb{P}$, we will assume, without loss of generality, that $\mathcal{S}_\infty$ is Harris recurrent [MT12, Ch. 9], since, by Qin [Qin23, Lem. 9], $\mathcal{S}_\infty$ is equal to a geometrically ergodic Harris chain with probability 1.

CLT for $\frac{1}{\sqrt{n}}\sum_{i=1}^n \xi_i$ We now show that $\frac{1}{\sqrt{n}}\sum_{i=1}^n \xi_i$ converges to a Gaussian random element taking values in $\mathcal{H}_{\boldsymbol{k}_\mathbb{Q}}$. We separate the proof in two parts: first when the initial distribution $\mu_1 = \mathbb{Q}$ and next when $\mu_1 \neq \mathbb{Q}$.

Case 1: $\mu_1 = \mathbb{Q}$ When $\mu_1 = \mathbb{Q}$, $\mathcal{S}_\infty$ is a strictly stationary chain. By Bradley [Bra05, Thm. 3.7 and (1.11)], since $\mathcal{S}_\infty$ is geometrically ergodic, its strong mixing coefficients $(\tilde{\alpha}_i)_{i\in\mathbb{N}}$ satisfy $\tilde{\alpha}_i \leq C\rho^i$ for some $C > 0$ and $\rho \in [0, 1)$ and all $i \in \mathbb{N}$. Since each ξ_i is a measurable function of x_i, the strong mixing coefficients $(\alpha_i)_{i\in\mathbb{N}}$ of $(\xi_i)_{i\in\mathbb{N}}$ satisfy $\alpha_i \leq \tilde{\alpha}_i \leq C\rho^i$ for each $i \in \mathbb{N}$. Consequently, $\sum_{i\in\mathbb{N}} i^{2/\delta}\alpha_i < \infty$ for $\delta = 2\mathfrak{q} - 2 > 0$. Note that we also have

$$\begin{aligned}\mathbb{E}_{z\sim\mathbb{Q}}[\|\boldsymbol{k}_\mathbb{Q}(z,\cdot)\|_{\boldsymbol{k}_\mathbb{Q}}^{2+\delta}] = \mathbb{E}_{z\sim\mathbb{Q}}[\boldsymbol{k}_\mathbb{Q}(z,z)^\mathfrak{q}] &= \mathbb{E}_{z\sim\mathbb{Q}}[\tfrac{\mathrm{d}\mathbb{P}}{\mathrm{d}\mathbb{Q}}(z)^{2\mathfrak{q}}\boldsymbol{k}_\mathbb{P}(z,z)^\mathfrak{q}] \\ &= \mathbb{E}_{x\sim\mathbb{P}}[\tfrac{\mathrm{d}\mathbb{P}}{\mathrm{d}\mathbb{Q}}(x)^{2\mathfrak{q}-1}\boldsymbol{k}_\mathbb{P}(x,x)^\mathfrak{q}] < \infty,\end{aligned}$$

$\mathbb{E}_{x_i\sim\mathbb{Q}}[\xi_i] = \mathbb{E}_{x_i\sim\mathbb{P}}[\boldsymbol{k}_\mathbb{P}(x_i,\cdot)] = 0$ and that $\mathcal{H}_{\boldsymbol{k}_\mathbb{Q}}$ is separable. Since $\mathcal{S}_\infty$ is a strictly stationary chain, we conclude that $(\xi_i)_{i\in\mathbb{N}}$ is a strictly stationary centered sequence of $\mathcal{H}_{\boldsymbol{k}_\mathbb{Q}}$-valued random variables satisfying the conditions needed to invoke Merlevède, Peligrad, and Utev [MPU97, Cor. 1], and hence $\sum_{i=1}^n \xi_i/\sqrt{n}$ converges in distribution to a Gaussian random element taking values in $\mathcal{H}_{\boldsymbol{k}_\mathbb{Q}}$.

Case 2: $\mu_1 \neq \mathbb{Q}$ Since $\mathcal{S}_\infty$ is positive Harris and $\sum_{i=1}^n \xi_i/\sqrt{n}$ satisfies a CLT for the initial distribution $\mu_1 = \mathbb{Q}$, Meyn and Tweedie [MT12, Prop. 17.1.6] implies that $\sum_{i=1}^n \xi_i/\sqrt{n}$ also satisfies the same CLT for any initial distribution μ_1.

Putting the pieces together for (3.21) Since, for any initial distribution for x_1, the sequence $(\sum_{i=1}^n \xi_i/\sqrt{n})_{n\in\mathbb{N}}$ converges in distribution and that $\mathcal{H}_{\boldsymbol{k}_\mathbb{Q}}$ is separable and (by virtue of being a Hilbert space) complete, Prokhorov's theorem [Bil13, Thm. 5.2] implies that the sequence is also tight, i.e., $\|\sum_{i=1}^n \xi_i\|_{\boldsymbol{k}_\mathbb{Q}}/\sqrt{n} = O_p(1)$. Consequently,

$$\|\tfrac{1}{n}\textstyle\sum_{i=1}^n \frac{\mathrm{d}\mathbb{P}}{\mathrm{d}\mathbb{Q}}(x_i)\boldsymbol{k}_\mathbb{P}(x_i,\cdot)\|_{\boldsymbol{k}_\mathbb{P}} \overset{(3.22)}{=} \|\tfrac{1}{n}\textstyle\sum_{i=1}^n \xi_i\|_{\boldsymbol{k}_\mathbb{Q}} = \frac{1}{\sqrt{n}}\cdot\|\frac{\sum_{i=1}^n \xi_i}{\sqrt{n}}\|_{\boldsymbol{k}_\mathbb{Q}} = O_p(n^{-\frac{1}{2}}),$$

as desired for the first claim in (3.21). Moreover, the strong law of large numbers for positive Harris chains [MT12, Thm. 17.0.1(i)] implies that $\frac{1}{n}\sum_{i\in[n]} \frac{\mathrm{d}\mathbb{P}}{\mathrm{d}\mathbb{Q}}(x_i)$ converges almost surely to $\mathbb{E}_{z\sim\mathbb{Q}}[\frac{\mathrm{d}\mathbb{P}}{\mathrm{d}\mathbb{Q}}(z)] = 1$ as desired for the second claim in (3.21). □

♢ 3.C.3 Proof of Thm. 3.2: Better-than-i.i.d. debiasing via simplex reweighting

We start with Thm. 3.C.1, proved in Sec. 3.C.4, that bounds $\mathrm{MMD}_{\mathrm{OPT}}$ in terms of the eigenvalues of the integral operator of the kernel $\boldsymbol{k}_\mathbb{P}$. Our proof makes use of a weight construction from Liu and Lee [LL17, Theorem 3.2], but is a non-trivial generalization of their proof as we no longer assume uniform bounds on the eigenfunctions, and instead leverage truncated variations of Bernstein's inequality (Lems. 3.C.2 and 3.C.3) to establish suitable concentration bounds.

Theorem 3.C.1 (Debiasing via i.i.d. simplex reweighting)**.** *Consider a kernel $\boldsymbol{k}_{\mathbb{P}}$ satisfying Assum. 3.1 with $\mathfrak{p} = 2$. Let $(\lambda_\ell)_{\ell=1}^{\infty}$ be the decreasing sequence of eigenvalues of $T_{\boldsymbol{k}_{\mathbb{P}},\mathbb{P}}$ defined in (3.6). Let $\mathcal{S}_n$ be a sequence of $n \in 2\mathbb{N}$ i.i.d. random variables with law $\mathbb{Q}$ such that $\mathbb{P}$ is absolutely continuous with respect to $\mathbb{Q}$ and $\|\frac{\mathrm{d}\mathbb{P}}{\mathrm{d}\mathbb{Q}}\|_\infty \leq M$ for some $M > 0$. Futhermore, assume there exist constants $\delta_n, B_n > 0$ such that $\Pr(\|\boldsymbol{k}_{\mathbb{P}}\|_n > B_n) < \delta_n$. Then for all $L \in \mathbb{N}$ such that $\lambda_L > 0$, we have*

$$\mathbb{E}[\mathrm{MMD}^2_{\boldsymbol{k}_{\mathbb{P}}}(\mathbb{S}_n^{w_{\mathrm{OPT}}}, \mathbb{P})] \leq \tfrac{8M}{n}\left(\tfrac{2M}{n}\tfrac{\mathbb{E}_{x\sim\mathbb{P}}[\boldsymbol{k}^2_{\mathbb{P}}(x,x)]}{\lambda_L} + \textstyle\sum_{\ell>L}\lambda_\ell\right) + \epsilon_n \mathbb{E}[\boldsymbol{k}^2_{\mathbb{P}}(x_1,x_1)], \tag{3.23}$$

where

$$\epsilon_n^2 \triangleq n\exp(\tfrac{-3n}{16MB_n/\lambda_L}) + 2\exp(\tfrac{-n}{16M^2}) + 2\exp(-\tfrac{n}{64M^2(\mathbb{E}_{x\sim\mathbb{P}}[\boldsymbol{k}_{\mathbb{P}}(x,x)]+B_n/12)/\lambda_L}) + \delta_n. \tag{3.24}$$

Given Thm. 3.C.1, Thm. 3.2 follows, i.e., we have $\mathbb{E}[\mathrm{MMD}^2_{\boldsymbol{k}_{\mathbb{P}}}(\mathbb{S}_n^{w_{\mathrm{OPT}}}, \mathbb{P})] = o(n^{-1})$, as long as we can show (i) $\mathbb{E}[\boldsymbol{k}^2_{\mathbb{P}}(x_1,x_1)] < \infty$, which in turn holds when $\mathfrak{q} > 3$ as assumed in Thm. 3.2, and (ii) find sequences $(B_n)_{n=1}^{\infty}$, $(\delta_n)_{n=1}^{\infty}$, and $(L_n)_{n=1}^{\infty}$ such that $\Pr(\|\boldsymbol{k}_{\mathbb{P}}\|_n > B_n) < \delta_n$ for all n and the following conditions are met:

(a) $\frac{\mathbb{E}_{x\sim\mathbb{P}}[\boldsymbol{k}^2_{\mathbb{P}}(x,x)]}{\lambda_{L_n}} = o(n)$;

(b) $\frac{B_n}{\lambda_{L_n}} = O(n^\beta)$, for some $\beta < 1$;

(c) $\sum_{\ell>L_n}\lambda_\ell = o(1)$;

(d) $\delta_n = o(n^{-2})$.

We now proceed to establish these conditions under the assumptions of Thm. 3.2.

Condition (d) By the de La Vallée Poussin Theorem [Cha15, Thm. 1.3] applied to the $\mathbb{Q}$-integrable function $x \mapsto \boldsymbol{k}_{\mathbb{P}}(x,x)^{\mathfrak{q}}$ (which is a uniformly integrable family with one function), there exists a convex increasing function G such that $\lim_{t\to\infty}\frac{G(t)}{t} = \infty$ and $\mathbb{E}[G(\boldsymbol{k}_{\mathbb{P}}(x_1,x_1)^{\mathfrak{q}})] < \infty$. Thus,

$$\begin{aligned}\Pr(\boldsymbol{k}_{\mathbb{P}}(x_1,x_1) > n^{3/\mathfrak{q}}) = \Pr(\boldsymbol{k}_{\mathbb{P}}(x_1,x_1)^{\mathfrak{q}} > n^3) &= \Pr(G(\boldsymbol{k}_{\mathbb{P}}(x_1,x_1))^{\mathfrak{q}} > G(n^3)) \\ &\leq \tfrac{\mathbb{E}[G(\boldsymbol{k}_{\mathbb{P}}(x_1,x_1))^{\mathfrak{q}}]}{G(n^3)} = o(n^{-3}),\end{aligned}$$

where the last step uses $\lim_{t\to\infty}\frac{G(t)}{t} = \infty$. Hence by the union bound,

$$\Pr(\|\boldsymbol{k}_{\mathbb{P}}\|_n > n^{3/\mathfrak{q}}) = \Pr(\max_{i\in[n]}\boldsymbol{k}_{\mathbb{P}}(x_i,x_i) > n^{3/\mathfrak{q}}) \leq n\Pr(\boldsymbol{k}_{\mathbb{P}}(x_1,x_1) > n^{3/\mathfrak{q}}) = o(n^{-2}).$$

Hence if we set $B_n = n^\tau$ for $\tau \triangleq 3/\mathfrak{q} < 1$, there exists $(\delta_n)_{n=1}^{\infty}$ such that $\delta_n = o(n^{-2})$. This fulfills (d) and that $\Pr(\|\boldsymbol{k}_{\mathbb{P}}\|_n > B_n) < \delta_n$.

To prove remaining conditions, without loss of generality, we assume that $\lambda_\ell > 0$ for all $\ell \in \mathbb{N}$, since otherwise we can choose L_n to be, for all n, the largest ℓ such that $\lambda_\ell > 0$. Then $\sum_{\ell > L_n} \lambda_{L_n} = 0$ and all other conditions are met.

Condition (c) If $L_n \to \infty$, then (c) is fulfilled since $\sum_\ell \lambda_\ell < \infty$, which follows from Lem. 3.B.1(d) and that

$$\begin{aligned}\sum_\ell \lambda_\ell = \sum_{\ell=1}^\infty \lambda_i \mathbb{E}_{x\sim\mathbb{P}}[\phi_\ell(x)^2] = \sum_{\ell=1}^\infty \lambda_i \mathbb{E}_{x\sim\mathbb{P}}[\phi_\ell(x)^2] &= \mathbb{E}_{x\sim\mathbb{P}}[\textstyle\sum_{\ell=1}^\infty \lambda_i \phi_\ell(x)^2] \\ &= \mathbb{E}_{x\sim\mathbb{P}}[\boldsymbol{k}_\mathbb{P}(x,x)] < \infty.\end{aligned}$$

Conditions (a) and (b) Note that the condition (a) is subsumed by (b) since $\mathbb{E}_{x\sim\mathbb{P}}[\boldsymbol{k}_\mathbb{P}^2(x,x)] < \infty$. It remains to choose $(L_n)_{n=1}^\infty$ to satisfy (b) such that $\lim_{n\to\infty} L_n = \infty$. Define $L_n \triangleq \max\{\ell \in \mathbb{N} : \lambda_\ell \geq n^{\frac{\tau-1}{2}}\}$. Then L_n is well-defined for $n \geq (\frac{1}{\lambda_1})^{\frac{2}{1-\tau}}$, since for such n we have $\lambda_1 \geq n^{\frac{\tau-1}{2}}$. Hence for $n \geq (\frac{1}{\lambda_1})^{\frac{2}{1-\tau}}$, we have

$$\frac{B_n}{\lambda_{L_n}} \leq \frac{n^\tau}{n^{\frac{\tau-1}{2}}} = n^{\frac{\tau+1}{2}},$$

so (b) is satisfied with $\beta = \frac{\tau+1}{2} < 1$. Since $\tau < 1$, L_n is non-decreasing in n and $n^{\frac{\tau-1}{2}}$ decreases to 0. Since each $\lambda_\ell > 0$, we therefore have $\lim_{n\to\infty} L_n = \infty$. □

♢ 3.C.4 Proof of Thm. 3.C.1: Debiasing via i.i.d. simplex reweighting

We will slowly build up towards proving Thm. 3.C.1. First notice $\mathbb{E}_{x\sim\mathbb{P}}[\boldsymbol{k}_\mathbb{P}^2(x,x)] < \infty$ implies $\mathbb{E}_{x\sim\mathbb{P}}[\boldsymbol{k}_\mathbb{P}(x,x)] < \infty$, so Lem. 3.B.1 holds. Fix any $L \in \mathbb{N}$ satisfying $\lambda_L > 0$. Since n is even, we can define $\mathcal{D}_0 \triangleq [n/2]$ and $\mathcal{D}_1 \triangleq [n] \setminus \mathcal{D}_0$. We will use $\mathcal{S}_{\mathcal{D}_0}$ and $\mathcal{S}_{\mathcal{D}_1}$ to denote the subsets of $\mathcal{S}_n$ with indices in $\mathcal{D}_0$ and $\mathcal{D}_1$ respectively. Let $(\phi_\ell)_{\ell=1}^\infty \subset \mathcal{H}_{\boldsymbol{k}_\mathbb{P}}$ be eigenfunctions corresponding to the eigenvalues $(\lambda_\ell)_{\ell=1}^\infty$ by Lem. 3.B.1(c), so that $(\phi_\ell)_{\ell=1}^\infty$ is an orthonormal system of $\mathcal{L}^2(\mathbb{P})$.

We start with a useful lemma.

Lemma 3.C.1 ($\mathcal{H}_{\boldsymbol{k}_\mathbb{P}}$ consists of mean-zero functions)**.** *Let $\boldsymbol{k}_\mathbb{P}$ be a kernel satisfying Assum. 3.1. Then for any $f \in \mathcal{H}_{\boldsymbol{k}_\mathbb{P}}$, we have $\mathbb{P}f = 0$.*

Proof. Fix $f \in \mathcal{H}_{\boldsymbol{k}_\mathbb{P}}$. By Steinwart and Christmann [SC08, Thm 4.26], f is $\mathbb{P}$ integrable. Consider the linear operator I that maps $f \mapsto \mathbb{P}f$. Since

$$\begin{aligned}|I(f)| = |\mathbb{P}f| \leq \mathbb{P}\,|f| = \int |\langle f, \boldsymbol{k}_\mathbb{P}(x,\cdot)\rangle_{\boldsymbol{k}_\mathbb{P}}|\,\mathrm{d}\mathbb{P} &\leq \int \|f\|_{\boldsymbol{k}_\mathbb{P}} \sqrt{\boldsymbol{k}_\mathbb{P}(x,x)}\mathrm{d}\mathbb{P} \\ &= \|f\|_{\boldsymbol{k}_\mathbb{P}}\, \mathbb{E}_{x\sim\mathbb{P}}[\boldsymbol{k}_\mathbb{P}(x,x)^{\frac{1}{2}}].\end{aligned}$$

Hence I is a continuous linear operator, so by the Riez representation theorem [SC08, Thm. A.5.12], there exists $g \in \mathcal{H}_{\boldsymbol{k}_\mathbb{P}}$ such that $I(h) = \langle h, g\rangle_{\boldsymbol{k}_\mathbb{P}}$ for any $h \in \mathcal{H}_{\boldsymbol{k}_\mathbb{P}}$.

By Steinwart and Christmann [SC08, Thm. 4.21], the set

$$H_{\text{pre}} \triangleq \left\{\sum_{i=1}^n \alpha_i \boldsymbol{k}_{\mathbb{P}}(\cdot, x_i) : n \in \mathbb{N}, (\alpha_i)_{i\in[n]} \subset \mathbb{R}, (x_i)_{i\in[n]} \subset \mathbb{R}^d\right\}$$

is dense in $\mathcal{H}_{\boldsymbol{k}_{\mathbb{P}}}$. Note that H_{pre} consists of mean zero functions under $\mathbb{P}$ by linearity. So there exists f_n converging to f in $\mathcal{H}_{\boldsymbol{k}_{\mathbb{P}}}$ where each f_n has $\mathbb{P}f_n = I(f_n) = \langle f_n, g\rangle_{\boldsymbol{k}_{\mathbb{P}}} = 0$. Since

$$\lim_{n\to\infty} |\langle f, g\rangle_{\boldsymbol{k}_{\mathbb{P}}} - \langle f_n, g\rangle_{\boldsymbol{k}_{\mathbb{P}}}| = \lim_{n\to\infty} |\langle f - f_n, g\rangle_{\boldsymbol{k}_{\mathbb{P}}}| \leq \lim_{n\to\infty} \|f - f_n\|_{\boldsymbol{k}_{\mathbb{P}}} \|g\|_{\boldsymbol{k}_{\mathbb{P}}} = 0,$$

we have $\mathbb{P}f = \langle f, g\rangle_{\boldsymbol{k}_{\mathbb{P}}} = 0$. □

In particular, the assumption $\mathbb{E}_{x\sim\mathbb{P}}[\boldsymbol{k}_{\mathbb{P}}^2(x,x)] < \infty$ of Thm. 3.C.1 implies $\mathbb{E}_{x\sim\mathbb{P}}[\boldsymbol{k}_{\mathbb{P}}(x,x)^{\frac{1}{2}}] < \infty$, so Lem. 3.C.1 holds.

Step 1. Build control variate weights

Fix any $L \geq 1$ and $h \in \mathcal{H}_{\boldsymbol{k}_{\mathbb{P}}}$, and let $\hat{h}_{\mathcal{D}_0}$ denote the eigen-expansion truncated approximation of h based on $\mathcal{D}_0$,

$$\hat{h}_{\mathcal{D}_0}(x) \triangleq \sum_{\ell=1}^L \hat{\beta}_{\ell,0}\phi_\ell(x) \quad \text{for} \quad \hat{\beta}_{\ell,0} \triangleq \frac{2}{n}\sum_{i\in\mathcal{D}_0} h(x_i)\phi_\ell(x_i)\xi(x_i).$$

Then

$$\mathbb{E}[\hat{\beta}_{\ell,0}] = \mathbb{E}\left[\frac{2}{n}\sum_{i\in\mathcal{D}_0} h(x_i)\phi_\ell(x_i)\xi(x_i)\right] = \langle h, \phi_\ell\rangle_{\mathcal{L}^2(\mathbb{P})}. \tag{3.25}$$

Next, define the control variate

$$\hat{Z}_0[h] = \frac{2}{n}\sum_{i\in D_1}\left(\xi(x_i)(h(x_i) - \hat{h}_{\mathcal{D}_0}(x_i))\right). \tag{3.26}$$

which satisfies

$$\mathbb{E}[\hat{Z}_0[h]] = \mathbb{E}_{x\sim\mathbb{P}}\left[h(x) - \sum_{\ell=1}^L \mathbb{E}[\hat{\beta}_{\ell,0}]\phi_\ell(x)\right] = 0, \tag{3.27}$$

since functions in $\mathcal{H}_{\boldsymbol{k}_{\mathbb{P}}}$ have mean 0 with respect to $\mathbb{P}$ (Lem. 3.C.1). Similarly, we define $\hat{Z}_1[h]$ by swapping $\mathcal{D}_0$ and $\mathcal{D}_1$. Then we form $\hat{Z}[h] \triangleq \frac{\hat{Z}_0[h]+\hat{Z}_1[h]}{2}$. We can rewrite $\hat{Z}[h]$ as a quadrature rule over $\mathcal{S}_n$ [LL17, Lemma B.6]

$$\hat{Z}[h] = \sum_{i\in[n]} w_i h(x_i), \tag{3.28}$$

where w_i is defined as (whose randomness depends on the randomness in $\mathcal{S}_n$)

$$w_i \triangleq \begin{cases} \frac{1}{n}\xi(x_i) - \frac{2}{n^2}\sum_{j\in\mathcal{D}_1}\xi(x_i)\xi(x_j)\langle\Phi_L(x_i), \Phi_L(x_j)\rangle, \forall i \in \mathcal{D}_0, \\ \frac{1}{n}\xi(x_i) - \frac{2}{n^2}\sum_{j\in\mathcal{D}_0}\xi(x_i)\xi(x_j)\langle\Phi_L(x_i), \Phi_L(x_j)\rangle, \forall i \in \mathcal{D}_1, \end{cases} \tag{3.29}$$

and $\Phi_L(x) \triangleq (\phi_1(x), \dots, \phi_L(x))$.

Step 2. Show $\mathbb{E}[\mathrm{MMD}^2_{\boldsymbol{k}_{\mathbb{P}}}(\mathbb{S}^w_n, \mathbb{P})] = o(n^{-1})$

We first bound the variance of the control variate $\hat{Z}_0[h]$ for $h = \phi_{\ell'}$ for $\ell' \in \mathbb{N}$. Let us fix $\ell' \in \mathbb{N}$. From (3.26), we compute

$$\begin{aligned}
\mathbb{E}[\hat{Z}_0[h]^2] &= \tfrac{4}{n^2}\mathbb{E}\left[\left(\textstyle\sum_{i\in\mathcal{D}_1} \xi(x_i)(h(x_i) - \hat{h}_{\mathcal{D}_0}(x_i))\right)^2\right] \\
&= \tfrac{4}{n^2}\mathbb{E}\left[\textstyle\sum_{i\in\mathcal{D}_1} \xi(x_i)^2(h(x_i) - \hat{h}_{\mathcal{D}_0}(x_i))^2\right] \\
&= \tfrac{2}{n}\mathbb{E}[\mathbb{E}_{x\sim\mathbb{Q}}[\xi(x)^2(h(x) - \hat{h}_{\mathcal{D}_0}(x))^2|\mathcal{S}_{\mathcal{D}_0}]] \\
&= \tfrac{2}{n}\mathbb{E}[\mathbb{E}_{x\sim\mathbb{P}}[\xi(x)(h(x) - \hat{h}_{\mathcal{D}_0}(x))^2|\mathcal{S}_{\mathcal{D}_0}]] \\
&\leq \tfrac{2M}{n}\mathbb{E}[\mathbb{E}_{x\sim\mathbb{P}}[(h(x) - \hat{h}_{\mathcal{D}_0}(x))^2|\mathcal{S}_{\mathcal{D}_0}]],
\end{aligned}$$

where in the second equality, the cross terms are zero due to the independence of points x_i and the equality (3.27). By the definition of $\hat{h}_{\mathcal{D}_0}$, we compute

$$\begin{aligned}
\mathbb{E}_{x\sim\mathbb{P}}[(h(x) - \hat{h}_{\mathcal{D}_0}(x))^2|\mathcal{S}_{\mathcal{D}_0}] &= \mathbb{E}_{x\sim\mathbb{P}}\left[\left(\phi_{\ell'}(x) - \textstyle\sum_{\ell\leq L} \hat{\beta}_{\ell,0}\phi_\ell(x)\right)^2\middle|\mathcal{S}_{\mathcal{D}_0}\right] \\
&= \mathbb{E}_{x\sim\mathbb{P}}\left[\phi^2_{\ell'}(x) + \textstyle\sum_{\ell\leq L} \hat{\beta}^2_{\ell,0}\phi^2_\ell(x) - 2\phi_{\ell'}(x)\sum_{\ell\leq L} \hat{\beta}_{\ell,0}\phi_\ell(x)\middle|\mathcal{S}_{\mathcal{D}_0}\right] \\
&= 1 + \textstyle\sum_{\ell\leq L} \hat{\beta}^2_{\ell,0} - 2\sum_{\ell\leq L} \hat{\beta}_{\ell,0}\mathbf{1}_{\ell'=\ell} \\
&= 1 + \textstyle\sum_{\ell\leq L} \hat{\beta}^2_{\ell,0} - 2\hat{\beta}_{\ell',0}\mathbf{1}_{\ell'\leq L},
\end{aligned}$$

where we use the fact that $(\phi_\ell)_{\ell=1}^\infty$ is an orthonormal system in $\mathcal{L}^2(\mathbb{P})$. By (3.25) with $h = \phi_{\ell'}$, we have $\mathbb{E}[\hat{\beta}_{\ell',0}] = 1$. On the other hand, we can bound, again using the orthonormality of $(\phi_\ell)_{\ell=1}^\infty$,

$$\begin{aligned}
\mathbb{E}[\hat{\beta}^2_{\ell,0}] = \mathbb{E}\left[\left(\tfrac{2}{n}\textstyle\sum_{i\in\mathcal{D}_0} \phi_\ell(x_i)\phi_{\ell'}(x_i)\xi(x_i)\right)^2\right] &= \tfrac{4}{n^2}\mathbb{E}\left[\textstyle\sum_{i\in\mathcal{D}_0}(\phi_\ell(x_i)\phi_{\ell'}(x_i)\xi(x_i))^2\right] \\
&\leq \tfrac{2M}{n}\mathbb{E}_{x\sim\mathbb{P}}[(\phi_\ell(x)\phi_{\ell'}(x))^2].
\end{aligned}$$

Thus for all $\ell' \in \mathbb{N}$,

$$\begin{aligned}
\mathbb{E}[\hat{Z}_0[\phi_{\ell'}]^2] &\leq \tfrac{2M}{n}\left(1 + \tfrac{2M}{n}\textstyle\sum_{\ell\leq L}\mathbb{E}_{x\sim\mathbb{P}}[(\phi_\ell(x)\phi_{\ell'}(x))^2] - 2\mathbf{1}_{\ell'\leq L}\right) \\
&\leq \tfrac{2M}{n}\left(\tfrac{2M}{n}\textstyle\sum_{\ell\leq L}\mathbb{E}_{x\sim\mathbb{P}}[(\phi_\ell(x)\phi_{\ell'}(x))^2] + \mathbf{1}_{\ell'>L}\right).
\end{aligned}$$

Since $\hat{Z}[h] = \frac{\hat{Z}_0[h]+\hat{Z}_1[h]}{2}$ and $(\frac{a+b}{2})^2 \leq \frac{a^2+b^2}{2}$ for $a, b \in \mathbb{R}$, and, by symmetry, $\mathbb{E}[\hat{Z}_0[h]^2] = \mathbb{E}[\hat{Z}_1[h]^2]$, we have

$$\mathbb{E}[\hat{Z}[\phi_{\ell'}]^2] \leq \tfrac{2M}{n}\left(\tfrac{2M}{n}\textstyle\sum_{\ell\leq L}\mathbb{E}_{x\sim\mathbb{P}}[(\phi_\ell(x)\phi_{\ell'}(x))^2] + \mathbf{1}_{\ell'>L}\right). \tag{3.30}$$

Now we have

$$\begin{aligned}\mathbb{E}[\mathrm{MMD}^2_{\boldsymbol{k}_\mathbb{P}}(\mathbb{S}^w_n,\mathbb{P})] &= \mathbb{E}\left[\textstyle\sum_{i,j\in[n]} w_i w_j \boldsymbol{k}_\mathbb{P}(x_i,x_j)\right]\\ &= \mathbb{E}\left[\textstyle\sum_{i,j\in[n]} w_i w_j \sum_{\ell'=1}^\infty \lambda_{\ell'}\phi_{\ell'}(x_i)\phi_{\ell'}(x_j)\right]\\ &= \mathbb{E}\left[\textstyle\sum_{\ell'=1}^\infty \sum_{i,j\in[n]} w_i w_j \lambda_{\ell'}\phi_{\ell'}(x_i)\phi_{\ell'}(x_j)\right]\\ &= \mathbb{E}\left[\textstyle\sum_{\ell'=1}^\infty \lambda_{\ell'}\left(\sum_{i\in[n]} w_i\phi_{\ell'}(x_i)\right)^2\right]\\ &= \textstyle\sum_{\ell'=1}^\infty \lambda_{\ell'}\mathbb{E}\left[\left(\sum_{i\in[n]} w_i\phi_{\ell'}(x_i)\right)^2\right] = \sum_{\ell'=1}^\infty \lambda_{\ell'}\mathbb{E}[\hat{Z}[\phi_{\ell'}]^2],\end{aligned}$$

where the second and third equalities are due to the absolute convergence of the Mercer series (Lem. 3.B.1(d)), the fourth equality follows from Tonelli's theorem [SC08, Thm. A.3.10], and the last step is due to (3.28). Plugging in (3.30), we have

$$\mathbb{E}[\mathrm{MMD}^2_{\boldsymbol{k}_\mathbb{P}}(\mathbb{S}^w_n,\mathbb{P})] \le \tfrac{2M}{n}\left(\tfrac{2M}{n}\textstyle\sum_{\ell'=1}^\infty\sum_{\ell\le L}\lambda_{\ell'}\mathbb{E}_{x\sim\mathbb{P}}[(\phi_\ell(x)\phi_{\ell'}(x))^2] + \sum_{\ell>L}\lambda_\ell\right).$$

Since the eigenvalues are non-negative and non-increasing, we can write, by (3.7),

$$\begin{aligned}\boldsymbol{k}^2_\mathbb{P}(x,x) &= \left(\textstyle\sum_{\ell=1}^\infty \lambda_\ell\phi_\ell(x)^2\right)^2 \ge \sum_{\ell'=1}^\infty\sum_{\ell\le L}\lambda_{\ell'}\lambda_\ell(\phi_\ell(x)\phi_{\ell'}(x))^2\\ &\ge \lambda_L\textstyle\sum_{\ell'=1}^\infty\sum_{\ell\le L}\lambda_{\ell'}(\phi_\ell(x)\phi_{\ell'}(x))^2.\end{aligned}$$

Thus by Tonelli's theorem [SC08, Thm. A.3.10],

$$\textstyle\sum_{\ell'=1}^\infty\sum_{\ell\le L}\lambda_{\ell'}\mathbb{E}_{x\sim\mathbb{P}}[(\phi_\ell(x)\phi_{\ell'}(x))^2] = \mathbb{E}_{x\sim\mathbb{P}}\left[\sum_{\ell'=1}^\infty\sum_{\ell\le L}\lambda_{\ell'}(\phi_\ell(x)\phi_{\ell'}(x))^2\right] \le \tfrac{\mathbb{E}_{x\sim\mathbb{P}}[\boldsymbol{k}^2_\mathbb{P}(x,x)]}{\lambda_L}.$$

Finally, we have

$$\mathbb{E}[\mathrm{MMD}^2_{\boldsymbol{k}_\mathbb{P}}(\mathbb{S}^w_n,\mathbb{P})] \le \tfrac{2M}{n}\left(\tfrac{2M}{n}\tfrac{\mathbb{E}_{x\sim\mathbb{P}}[\boldsymbol{k}^2_\mathbb{P}(x,x)]}{\lambda_L} + \textstyle\sum_{\ell>L}\lambda_\ell\right). \tag{3.31}$$

Step 3. Meet the non-negative constraint

We now show that the weights (3.29) are nonnegative and sum close to one with high probability. For $i\in\mathcal{D}_0$, we have

$$w_i = \tfrac{1}{n}\xi(x_i)\,(1-T_i) \quad \text{for} \quad T_i \triangleq \tfrac{2}{n}\textstyle\sum_{j\in\mathcal{D}_1}\xi(x_j)\langle\Phi_L(x_i),\Phi_L(x_j)\rangle.$$

Our first goal is to derive an upper bound for $\Pr\left(\min_{i\in\mathcal{D}_0} w_i < 0\right)$. Define the event

$$E_n \triangleq \left\{\|\boldsymbol{k}_\mathbb{P}\|_n \le B_n\right\}, \tag{3.32}$$

so $\Pr(E_n^c) < \delta_n$ by the assumption on $\|\boldsymbol{k}_\mathbb{P}\|_n$. Then

$$\Pr\left(\min_{i\in[n]} w_i < 0, E_n\right) = \Pr\left(\max_{i\in[n]} T_i > 1, E_n\right) \le n\Pr(T_1\mathbf{1}_{E_n} > 1), \tag{3.33}$$

where we applied the union bound and used the fact that $T_i\mathbf{1}_{E_n}$ has the same law for different i. To further bound $\Pr(T_1\mathbf{1}_{E_n} > 1)$, we will use the following lemma.

Lemma 3.C.2 (Truncated Bernstein inequality)**.** *Let $X_1, \dots, X_n$ be i.i.d. random variables with $\mathbb{E}[X_1] = 0$ and $\mathbb{E}[X_1^2] < \infty$. For any $B > 0$, $t > 0$,*

$$\Pr\left(\tfrac{1}{n}\textstyle\sum_{i\in[n]} X_i \mathbf{1}_{X_i \le B} > t\right) \le \exp\left(\frac{-nt^2}{2(\mathbb{E}[X_1^2] + \frac{Bt}{3})}\right).$$

Proof of Lem. 3.C.2. Fix any $B > 0$ and $t > 0$ and define, for each $i \in [n]$, $Y_i \triangleq X_i \mathbf{1}_{X_i \le B}$. Then $Y_i \le B$,

$$\begin{aligned}\mathbb{E}[Y_i] &= \mathbb{E}[X_i \mathbf{1}_{X_i \le B}] \le \mathbb{E}[X_i \mathbf{1}_{X_i \le B}] + \mathbb{E}[X_i \mathbf{1}_{X_i > B}] = \mathbb{E}[X_i] = 0, \quad \text{and}\\ \mathbb{E}[Y_i^2] &= \mathbb{E}[X_i^2 \mathbf{1}_{X_i \le B}] \le \mathbb{E}[X_i^2] = \mathbb{E}[X_1^2].\end{aligned}$$

Now we can invoke the non-positivity of $\mathbb{E}[Y_i]$, the one-sided Bernstein inequality [Wai19, Prop. 2.14], and the relation $\mathbb{E}[Y_i^2] \le \mathbb{E}[X_1^2]$ to conclude that

$$\begin{aligned}\Pr\left(\tfrac{1}{n}\textstyle\sum_{i\in[n]} Y_i > t\right) &\le \Pr\left(\tfrac{1}{n}\textstyle\sum_{i\in[n]} (Y_i - \mathbb{E}[Y_i]) > t\right)\\ &\le \exp\left(\frac{-nt^2}{2(\frac{1}{n}\sum_{i\in[n]} \mathbb{E}[Y_i^2] + \frac{Bt}{3})}\right)\\ &\le \exp\left(\frac{-nt^2}{2(\mathbb{E}[X_1^2] + \frac{Bt}{3})}\right).\end{aligned}$$

□

For $j \in \mathcal{D}_1$, define $X_j \triangleq \xi(x_j)\langle \Phi_L(x_1), \Phi_L(x_j)\rangle$ and note that

$$\begin{aligned}\mathbb{E}[X_j | x_1] &= \mathbb{E}_{x\sim\mathbb{Q}}[\xi(x)\langle \Phi_L(x_1), \Phi_L(x)\rangle | x_1] = \mathbb{E}_{x\sim\mathbb{P}}[\langle \Phi_L(x_1), \Phi_L(x)\rangle | x_1] = 0\\ \mathbb{E}[X_j^2 | x_1] &= \mathbb{E}[\xi(x_j)^2 \langle \Phi_L(x_1), \Phi_L(x_j)\rangle^2 | x_1] \le M \mathbb{E}_{x\sim\mathbb{P}}[\langle \Phi_L(x_1), \Phi_L(x)\rangle^2 | x_1]\\ &= M \mathbb{E}_{x\sim\mathbb{P}}\left[\textstyle\sum_{\ell,\ell' \le L} \phi_\ell(x_1)\phi_{\ell'}(x_1)\phi_\ell(x)\phi_{\ell'}(x) \Big| x_1\right]\\ &= M \textstyle\sum_{\ell,\ell' \le L} \phi_\ell(x_1)\phi_{\ell'}(x_1)\mathbb{E}_{x\sim\mathbb{P}}\left[\phi_\ell(x)\phi_{\ell'}(x)\right]\\ &= M \left\|\Phi_L(x_1)\right\|_2^2.\end{aligned}$$

Since $\lambda_1 \ge \lambda_2 \ge \cdots \ge 0$, for any $x \in \mathbb{R}^d$, we can bound $\|\Phi_L(x)\|_2^2$ via

$$\|\Phi_L(x)\|_2^2 = \textstyle\sum_{\ell \le L} \phi_\ell(x)^2 \le \frac{\sum_{\ell\le L} \lambda_\ell \phi_\ell(x)^2}{\lambda_L} \le \frac{\sum_{\ell=1}^\infty \lambda_\ell \phi_\ell(x)^2}{\lambda_L} = \frac{\boldsymbol{k}_{\mathbb{P}}(x,x)}{\lambda_L}, \tag{3.34}$$

where we applied Lem. 3.B.1(d) for the last equality. Thus

$$|X_j| \le M \left\|\Phi_L(x_1)\right\|_2 \left\|\Phi_L(x_j)\right\|_2 \le M \left\|\Phi_L(x_1)\right\|_2 \sqrt{\frac{\boldsymbol{k}_{\mathbb{P}}(x_j,x_j)}{\lambda_L}},$$

so if we let $B \triangleq \sqrt{\frac{B_n}{\lambda_L}} M \left\|\Phi_L(x_1)\right\|_2$, then

$$E_n = \left\{\sup_{i\in[n]} \boldsymbol{k}_{\mathbb{P}}(x_i, x_i) \le B_n\right\} \subset \textstyle\bigcap_{j\in\mathcal{D}_1} \{|X_j| \le B\}.$$

Since $T_1 = \frac{2}{n}\sum_{j\in\mathcal{D}_1} X_j$, we have inclusions of events

$$\{T_1\mathbf{1}_{E_n} > 1\} = \{T_1 > 1\} \cap E_n \subset \left\{\tfrac{2}{n}\textstyle\sum_{j\in\mathcal{D}_1} X_j\mathbf{1}_{X_j\le B} > 1\right\}.$$

Thus Lem. 3.C.2 with $t = 1$ and conditioned on x_1 implies

$$\begin{aligned}\Pr\left(T_1\mathbf{1}_{E_n} > 1|x_1\right) &\le \Pr\left(\tfrac{2}{n}\textstyle\sum_{j\in\mathcal{D}_1} X_j\mathbf{1}_{X_j\le B} > 1\Big|x_1\right)\\ &\le \exp\left(\frac{-n}{4(M\|\Phi_L(x_1)\|_2^2+\sqrt{\frac{B_n}{\lambda_L}}M\|\Phi_L(x_1)\|_2/3)}\right).\end{aligned}$$

On event $\{\boldsymbol{k}_{\mathbb{P}}(x_1, x_1) \le B_n\}$, by (3.34), we have

$$\|\Phi_L(x_1)\|_2 \le \sqrt{\tfrac{B_n}{\lambda_L}}.$$

Hence

$$\Pr\left(T_1\mathbf{1}_{E_n} > 1|x_1\right)\mathbf{1}_{\boldsymbol{k}_{\mathbb{P}}(x_1,x_1)\le B_n} \le \exp\left(\frac{-n}{\frac{16}{3}M\frac{B_n}{\lambda_L}}\right).$$

On the other hand, $\{\boldsymbol{k}_{\mathbb{P}}(x_1, x_1) > B_n\} \notin E_n$, so

$$\Pr\left(T_1\mathbf{1}_{E_n} > 1|x_1\right)\mathbf{1}_{\boldsymbol{k}_{\mathbb{P}}(x_1,x_1)> B_n} = 0$$

Thus

$$\Pr(T_1\mathbf{1}_{E_n} > 1) = \mathbb{E}[\Pr(T_1\mathbf{1}_{E_n} > 1|x_1)] \le \exp\left(\frac{-n}{\frac{16}{3}M\frac{B_n}{\lambda_L}}\right).$$

Combining the last inequality with (3.33), we have:

$$\Pr\left(\min_{i\in[n]} w_i < 0, E_n\right) \le n\exp\left(\frac{-n}{\frac{16}{3}M\frac{B_n}{\lambda_L}}\right). \tag{3.35}$$

Step 4. Meet the sum-to-one constraint
Let

$$S \triangleq \textstyle\sum_{i\in\mathcal{D}_0} w_i = \sum_{i\in\mathcal{D}_0}\frac{1}{n}\xi(x_i)\left(1 - \frac{2}{n}\sum_{j\in\mathcal{D}_1}\xi(x_j)\langle\Phi_L(x_i), \Phi_L(x_j)\rangle\right).$$

We now derive a bound for $\Pr(S < 1/2 - t/2)$ for $t \in (0, 1)$. Let

$$S_1 \triangleq \tfrac{1}{n}\textstyle\sum_{i\in\mathcal{D}_0}\xi(x_i), \quad S_2 \triangleq -\frac{2}{n^2}\sum_{i\in\mathcal{D}_0}\sum_{j\in\mathcal{D}_1}\xi(x_i)\xi(x_j)\langle\Phi_L(x_i), \Phi_L(x_j)\rangle,$$

so $S = S_1 + S_2$. Note that $\mathbb{E}[S_1] = 1/2$ and $\mathbb{E}[S_2] = 0$ since $\mathcal{D}_0$ and $\mathcal{D}_1$ are disjoint. Let E_n be the same event defined as in (3.32). For $t_1 \in (0, t/2)$ to be determined later and $t_2 \triangleq t/2 - t_1$, we have, by the union bound

$$\Pr(S < 1/2 - t/2, E_n) \leq \Pr(S_1 < 1/2 - t_1, E_n) + \Pr(S_2 < -t_2, E_n).$$

By Hoeffding's inequality and the assumption $\xi(x) \leq M$, we have

$$\Pr(S_1 < 1/2 - t_1, E_n) \leq \Pr\left(\frac{2}{n}\sum_{i\in\mathcal{D}_0}\frac{\xi(x_i)}{2} - 1/2 < -t_1\right) \leq \exp\left(\frac{-2(n/2)t_1^2}{(M/2)^2}\right) = \exp\left(\frac{-4nt_1^2}{M^2}\right). \tag{3.36}$$

To give a concentration bound for $\Pr(S_2 < -t_2, E_n)$, we will use the following lemma.

Lemma 3.C.3 (U-statistic Bernstein's inequality)**.** *Let $h : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a function bounded above by $b > 0$. Assume $n \in 2\mathbb{N}$ and let $x_1, \ldots, x_n$ be i.i.d. random variables taking values in $\mathcal{X}$. Denote $m_h \triangleq \mathbb{E}[h(x_1, x_2)]$ and $\sigma_h^2 \triangleq \mathrm{Var}[h(x_1, x_2)]$. Let $\mathcal{D}_0 = [n/2]$ and $\mathcal{D}_1 = [n] \setminus [n/2]$. Define*

$$U \triangleq \frac{1}{(n/2)^2}\sum_{i\in\mathcal{D}_0}\sum_{j\in\mathcal{D}_1} h(x_i, x_j).$$

Then

$$\Pr(U - m_h > t) \leq \exp\left(\frac{-nt^2}{4(\sigma_h^2 + \frac{bt}{3})}\right).$$

Proof of Lem. 3.C.3. We adapt the proof from Pitcan [Pit17, Section 3] as follows. Let $k \triangleq n/2$. Define $V : \mathcal{X}^n \to \mathbb{R}$ as

$$V(x_1, \ldots, x_n) \triangleq \frac{1}{k}\sum_{i\in[k]} h(x_i, x_{i+k}).$$

Then note that

$$U = \frac{1}{k!}\sum_{\sigma\in\mathrm{perm}(k)} V_\sigma,$$
$$V_\sigma \triangleq V(x_{\sigma_1}, \ldots, x_{\sigma_k}),$$

where $\mathrm{perm}(k)$ is the set of all permutations of $[k]$; this is because every $h(x_i, x_j)$ term for $i \in \mathcal{D}_0, j \in \mathcal{D}_1$ will appear in the summation an equal number of times. For a fixed $\sigma \in \mathrm{perm}(k)$, the random variable $V(x_{\sigma_1}, \ldots, x_{\sigma_k}, x_{k+1}, \ldots, x_n)$ is a sum of k i.i.d. terms $h(x_{\sigma_i}, x_{i+k})$. Denote $V = V(x_1, \ldots, x_n)$. For any $s > 0$, we have, by independence,

$$\begin{aligned}\mathbb{E}[e^{s(V-m_h)}] &= \mathbb{E}\left[\exp(\tfrac{s}{k}\textstyle\sum_{i\in[k]}(h(x_i, x_{i+[k]}) - m_h))\right] \\ &= \left(\mathbb{E}\left[\exp(\tfrac{s}{k}(h(x_1, x_2) - m_h))\right]\right)^k\end{aligned}$$

By the one-sided Bernstein's lemma Wainwright [Wai19, Prop. 2.14] applied to $\frac{h(x_1,x_2)}{k}$ which is upper bounded by $\frac{b}{k}$ with variance $\frac{\sigma_h^2}{k^2}$, we have

$$\mathbb{E}\left[\exp(s\tfrac{h(x_1,x_2)-m_h}{k})\right] \le \exp(\tfrac{s^2\sigma_h^2/2}{k(k-\frac{bs}{3})}),$$

for $s \in [0, 3k/b)$. Next, by Markov's inequality and Jensen's inequality,

$$\begin{aligned}
\Pr(U - m_h > t) &= \Pr(e^{s(U-m_h)} > e^{st}) \le \mathbb{E}[e^{s(U-m_h)}]e^{-st} \\
&= \mathbb{E}\left[\exp(\tfrac{1}{(n/2)!}\textstyle\sum_{\sigma\in\mathrm{perm}(n/2)} s(V_\sigma - m_h))\right] e^{-st} \\
&\le \mathbb{E}\left[\tfrac{1}{(n/2)!}\textstyle\sum_{\sigma\in\mathrm{perm}(n/2)} \exp(s(V_\sigma - m_h))\right] e^{-st} \\
&= \mathbb{E}[e^{s(V-m_h)}]e^{-st}.
\end{aligned}$$

Therefore,

$$\Pr(U - m_h > t) \le \exp(\tfrac{s^2\sigma_h^2}{2(k-\frac{bs}{3})} - st).$$

Now, we get the desired bound if we pick $s = \frac{k^2 t}{k\sigma_h^2 + \frac{ktb}{3}} \in [0, 3k/b)$ and simplify. □

Let

$$\begin{aligned}
h(x, x') &\triangleq \xi(x)\xi(x')\langle\Phi_L(x), \Phi_L(x')\rangle \\
\bar{h}(x, x') &\triangleq h(x, x')\mathbf{1}_{h(x,x')\le M^2\frac{B_n}{\lambda_L}}.
\end{aligned}$$

Then

$$\begin{aligned}
\Pr(S_2 < -t_2, E_n) &= \Pr\left(\tfrac{1}{(n/2)^2}\textstyle\sum_{i\in\mathcal{D}_0}\sum_{j\in\mathcal{D}_1} h(x_i, x_j) > 2t_2, E_n\right) \\
&\le \Pr\left(\tfrac{1}{(n/2)^2}\textstyle\sum_{i\in\mathcal{D}_0}\sum_{j\in\mathcal{D}_1} \bar{h}(x_i, x_j) > 2t_2\right), \qquad (3.37)
\end{aligned}$$

where the last inequality used the fact that, for $i \in \mathcal{D}_0, j \in \mathcal{D}_1$,

$$E_n \subset \{\max(\boldsymbol{k}_{\mathbb{P}}(x_i, x_i), \boldsymbol{k}_{\mathbb{P}}(x_j, x_j)) \le B_n\} \subset \left\{h(x_i, x_j) \le M^2\tfrac{B_n}{\lambda_L}\right\},$$

using (3.34). We further compute

$$\begin{aligned}
m_{\bar{h}} &= \mathbb{E}[\bar{h}(x_1, x_2)] \le \mathbb{E}[h(x_1, x_2)] = \mathbb{E}[\xi(x_1)\xi(x_2)\langle\Phi_L(x_1), \Phi_L(x_2)\rangle] \\
&= \textstyle\sum_{\ell\le L} \mathbb{E}[\xi(x_1)\xi(x_2)\phi_\ell(x_1)\phi_\ell(x_2)] \\
&= \textstyle\sum_{\ell\le L} (\mathbb{E}_{x\sim\mathbb{P}}[\phi_\ell(x)])^2 = 0,
\end{aligned}$$

and

$$\begin{aligned}\sigma_{\bar{h}}^2 &= \mathrm{Var}[\bar{h}(x_1,x_2)] \leq \mathbb{E}[\bar{h}(x_1,x_2)^2] \leq \mathbb{E}[h(x_1,x_2)^2] \\ &= \mathbb{E}\left[(\xi(x_1)\xi(x_2)\langle\Phi_L(x_1),\Phi_L(x_2)\rangle)^2\right] \\ &\leq M^2\mathbb{E}_{(x,x')\sim\mathbb{P}\times\mathbb{P}}[\langle\Phi_L(x),\Phi_L(x')\rangle^2] \\ &= M^2\mathbb{E}_{(x,x')\sim\mathbb{P}\times\mathbb{P}}\left[\textstyle\sum_{\ell,\ell'\leq L}\phi_\ell(x)\phi_{\ell'}(x)\phi_\ell(x')\phi_{\ell'}(x')\right] \\ &= M^2\textstyle\sum_{\ell,\ell'\leq L}(\mathbb{E}[\phi_\ell(x)\phi_{\ell'}(x)])^2 = LM^2.\end{aligned}$$

Since $\mathbb{E}_{x\sim\mathbb{P}}[\boldsymbol{k}_{\mathbb{P}}(x,x)] = \sum_\ell \lambda_\ell \geq L\lambda_L$, we have $L \leq \frac{\|\boldsymbol{k}_{\mathbb{P}}\|^2_{\mathcal{L}^2(\mathbb{P})}}{\lambda_L}$, so that $\sigma_{\bar{h}}^2 \leq \frac{M^2\|\boldsymbol{k}_{\mathbb{P}}\|^2_{\mathcal{L}^2(\mathbb{P})}}{\lambda_L}$. Applying Lem. 3.C.3 to $\bar{h}$, which is bounded by $M^2\frac{B_n}{\lambda_L}$ and using the fact that $m_{\bar{h}} \leq 0$, we have

$$\begin{aligned}\Pr\left(\tfrac{1}{(n/2)^2}\textstyle\sum_{i\in\mathcal{D}_0}\sum_{j\in\mathcal{D}_1}\bar{h}(x_i,x_j) > 2t_2\right) &\leq \Pr\left(\tfrac{1}{(n/2)^2}\textstyle\sum_{i\in\mathcal{D}_0}\sum_{j\in\mathcal{D}_1}\bar{h}(x_i,x_j) - m_{\bar{h}} > 2t_2\right) \\ &\leq \exp\left(\frac{-n(2t_2)^2}{4\left(\frac{M^2\|\boldsymbol{k}_{\mathbb{P}}\|^2_{\mathcal{L}^2(\mathbb{P})}}{\lambda_L} + 2M^2\frac{B_n}{\lambda_L}t_2/3\right)}\right). \qquad (3.38)\end{aligned}$$

Thus combining (3.36), (3.37), (3.38), we get

$$\Pr(S < 1/2 - t/2, E_n) \leq \exp\left(\tfrac{-4nt_1^2}{M^2}\right) + \exp\left(\frac{-nt_2^2}{\left(\frac{M^2\|\boldsymbol{k}_{\mathbb{P}}\|^2_{\mathcal{L}^2(\mathbb{P})}}{\lambda_L} + 2M^2\frac{B_n}{\lambda_L}t_2/3\right)}\right).$$

Finally, by symmetry and the union bound, for $t \in (0,1)$, $t \in (0,t/2)$ and $t_2 = t/2 - t_1$, we have

$$\begin{aligned}\Pr\left(\textstyle\sum_{i\in[n]} w_i < 1-t, E_n\right) &\leq \Pr\left(\textstyle\sum_{i\in\mathcal{D}_0} w_i < 1/2 - t/2, E_n\right) + \Pr\left(\textstyle\sum_{i\in\mathcal{D}_1} w_i < 1/2 - t/2, E_n\right) \\ &= 2\Pr(S < 1/2 - t/2, E_n) \\ &\leq 2\left(\exp\left(\tfrac{-4nt_1^2}{M^2}\right) + \exp\left(\frac{-nt_2^2}{\left(\frac{M^2\|\boldsymbol{k}_{\mathbb{P}}\|^2_{\mathcal{L}^2(\mathbb{P})}}{\lambda_L} + 2M^2\frac{B_n}{\lambda_L}t_2/3\right)}\right)\right). \qquad (3.39)\end{aligned}$$

Step 5. Putting it all together

Define the event

$$F_n = \left\{\min_{i\in[n]} w_i \geq 0, \textstyle\sum_{i\in[n]} w_i \geq \frac{1}{2}\right\}.$$

Then, by the union bound,

$$\Pr(F_n^c) \leq \Pr(\min_{i\in[n]} w_i < 0, E_n) + \Pr(\textstyle\sum_{i\in[n]} w_i < \frac{1}{2}, E_n) + \Pr(E_n^c).$$

Applying (3.35) and (3.39) to bound the last expression with $t = 1/2$, $t_1 = t_2 = 1/8$, we have $\Pr(F_n^c) \leq \epsilon_n^2$ for ϵ_n defined in (3.24). On the event F_n, if we define $w^+ \in \Delta_{n-1}$ via

$$w_i^+ \triangleq \frac{w_i}{\sum_{i\in[n]} w_i},$$

then $w_i^+ = \alpha w_i$ for $i \in [n]$ and $\alpha \triangleq \frac{1}{\sum_{i\in[n]} w_i} \leq 2$. Let $\tilde{w} \in \Delta_{n-1}$ be the weight defined by $\tilde{w}_1 = 1$ and $\tilde{w}_i = 0$ for $i > 1$.

Since w_{OPT} is the best simplex weight, we have

$$\mathrm{MMD}^2_{\boldsymbol{k}_{\mathbb{P}}}(\mathbb{S}_n^{w_{\mathrm{OPT}}}, \mathbb{P}) \leq \min(\mathrm{MMD}^2_{\boldsymbol{k}_{\mathbb{P}}}(\mathbb{S}_n^{w^+}, \mathbb{P}), \mathrm{MMD}^2_{\boldsymbol{k}_{\mathbb{P}}}(\mathbb{S}_n^{\tilde{w}}, \mathbb{P})).$$

Hence

$$\begin{aligned}\mathbb{E}\left[\mathrm{MMD}^2_{\boldsymbol{k}_{\mathbb{P}}}(\mathbb{S}_n^{w_{\mathrm{OPT}}}, \mathbb{P})\right] &= \mathbb{E}\left[\mathrm{MMD}^2_{\boldsymbol{k}_{\mathbb{P}}}(\mathbb{S}_n^{w_{\mathrm{OPT}}}, \mathbb{P})\mathbf{1}_{F_n}\right] + \mathbb{E}\left[\mathrm{MMD}^2_{\boldsymbol{k}_{\mathbb{P}}}(\mathbb{S}_n^{w_{\mathrm{OPT}}}, \mathbb{P})\mathbf{1}_{F_n^c}\right] \\ &\leq \mathbb{E}\left[\mathrm{MMD}^2_{\boldsymbol{k}_{\mathbb{P}}}(\mathbb{S}_n^{w^+}, \mathbb{P})\mathbf{1}_{F_n}\right] + \mathbb{E}\left[\mathrm{MMD}^2_{\boldsymbol{k}_{\mathbb{P}}}(\mathbb{S}_n^{\tilde{w}}, \mathbb{P})\mathbf{1}_{F_n^c}\right].\end{aligned}$$

For the first term, we have the bound

$$\begin{aligned}\mathbb{E}\left[\mathrm{MMD}^2_{\boldsymbol{k}_{\mathbb{P}}}(\mathbb{S}_n^{w^+}, \mathbb{P})\mathbf{1}_{F_n}\right] &= \mathbb{E}\left[\textstyle\sum_{i,j\in[n]} w_i^+ w_j^+ \boldsymbol{k}_{\mathbb{P}}(x_i, x_j)\mathbf{1}_{F_n}\right] = \mathbb{E}\left[\alpha^2 \textstyle\sum_{i,j\in[n]} w_i w_j \boldsymbol{k}_{\mathbb{P}}(x_i, x_j)\mathbf{1}_{F_n}\right] \\ &\leq 4\mathbb{E}\left[\textstyle\sum_{i,j\in[n]} w_i w_j \boldsymbol{k}_{\mathbb{P}}(x_i, x_j)\right] \leq \frac{8M}{n}\left(\frac{2M}{n}\frac{\mathbb{E}_{x\sim\mathbb{P}}[\boldsymbol{k}_{\mathbb{P}}^2(x,x)]}{\lambda_L} + \textstyle\sum_{\ell>L}\lambda_\ell\right),\end{aligned}$$

where we applied (3.31) for the last inequality. For the second term, by the Cauchy-Schwartz inequality,

$$\begin{aligned}\mathbb{E}\left[\mathrm{MMD}^2_{\boldsymbol{k}_{\mathbb{P}}}(\mathbb{S}_n^{\tilde{w}}, \mathbb{P})\mathbf{1}_{F_n^c}\right] &\leq \sqrt{\Pr(F_n^c)}\sqrt{\mathbb{E}\left[\left(\textstyle\sum_{i,j\in[n]} \boldsymbol{k}_{\mathbb{P}}(x_i, x_j)\tilde{w}_i\tilde{w}_j\right)^2\right]} \\ &\leq \sqrt{\Pr(F_n^c)}\sqrt{\mathbb{E}[\boldsymbol{k}_{\mathbb{P}}(x_1, x_1)^2]}.\end{aligned}$$

Putting everything together we obtain (3.23). □

■ 3.D Stein Kernel Thinning

In this section, we detail our Stein thinning implementation in Sec. 3.D.1, our kernel thinning implementation and analysis in Sec. 3.D.2, and our proof of Thm. 3.3 in Sec. 3.D.3.

♢ 3.D.1 Stein Thinning with sufficient statistics

For an input point set of size n, the original implementation of Stein Thinning of Riabiz et al. [Ria+22] takes $O(nm^2)$ time to output a coreset of size m. In Alg. 3.D.1, we show that this runtime can be improved to $O(nm)$ using sufficient statistics. The idea is to maintain a vector $g \in \mathbb{R}^n$ such that $g = 2\boldsymbol{k}_{\mathbb{P}}(\mathbb{S}_n, \mathbb{S}_n)w$ where w is the weight representing the current coreset.

Algorithm 3.D.1 SteinThinning (ST) with sufficient statistics

Input: kernel $\boldsymbol{k}_{\mathbb{P}}$ with zero-mean under $\mathbb{P}$, input points $\mathcal{S}_n = (x_i)_{i\in[n]}$, output size m
$w \leftarrow \mathbf{0} \in \mathbb{R}^n$
$j \leftarrow \arg\min_{i\in[n]} \boldsymbol{k}_{\mathbb{P}}(x_i, x_i)$
$w_j \leftarrow 1$
$g \leftarrow 2\boldsymbol{k}_{\mathbb{P}}(\mathcal{S}_n, x_j)$ ▷ *maintain sufficient statistics* $g = 2\boldsymbol{k}_{\mathbb{P}}(\mathcal{S}_n, \mathcal{S}_n)w$
for $t = 1$ **to** $m - 1$ **do**
  $j \leftarrow \arg\min_{i\in[n]}\{tg_i + \boldsymbol{k}_{\mathbb{P}}(x_i, x_i)\}$
  $w \leftarrow \frac{t}{t+1}w + \frac{1}{t+1}e_j$
  $g \leftarrow \frac{t}{t+1}g + \frac{2}{t+1}\boldsymbol{k}_{\mathbb{P}}(\mathcal{S}_n, x_j)$
end for
Return: w

♢ 3.D.2 Kernel Thinning targeting $\mathbb{P}$

Algorithm 3.D.2 KernelThinning (KT) (adapted from Dwivedi and Mackey [DM22b, Alg. 1])

Input: kernel $\boldsymbol{k}_{\mathbb{P}}$ with zero-mean under $\mathbb{P}$, input points $\mathcal{S}_n = (x_i)_{i\in[n]}$, multiplicity n' with $\log_2 \frac{n'}{m} \in \mathbb{N}$, weight $w \in \Delta_{n-1} \cap (\frac{\mathbb{N}_0}{n'})^n$, output size m with $\frac{n'}{m} \in 2^{\mathbb{N}}$, failure probability δ
$\mathtt{S} \leftarrow$ index sequence where $k \in [n]$ appears $n'w_k$ times
$\mathfrak{t} \leftarrow \log_2 \frac{n'}{m} \in \mathbb{N}$
$(\mathtt{I}^{(\ell)})_{\ell\in[2^{\mathfrak{t}}]} \leftarrow$ KT-SPLIT$(\boldsymbol{k}_{\mathbb{P}}, \mathcal{S}_n[\mathtt{S}], \mathfrak{t}, \delta/n')$ ▷ *KT-SPLIT is from Dwivedi and Mackey [DM22b, Algorithm 1a] and we set* $\delta_i = \delta$ *for all* i
$\mathtt{I}^{(\ell)} \leftarrow \mathtt{S}[\mathtt{I}^{(\ell)}]$ for each $\ell \in [2^{\mathfrak{t}}]$
$\mathtt{I} \leftarrow$ KT-Swap$(\boldsymbol{k}_{\mathbb{P}}, \mathcal{S}_n, (\mathtt{I}^{(\ell)})_{\ell\in[2^{\mathfrak{t}}]})$
$w_{\mathrm{KT}} \leftarrow$ simplex weight corresponding to $\mathtt{I}$ ▷ $w_i = \frac{\textit{number of occurrences of } i \textit{ in } \mathtt{I}}{|\mathtt{I}|}$
Return: $w_{\mathrm{KT}} \in \Delta_{n-1} \cap (\frac{\mathbb{N}_0}{m})^n$ ▷ *Hence* $\|w_{\mathrm{KT}}\|_0 \leq m$

Algorithm 3.D.3 KT-Swap (modified Dwivedi and Mackey [DM22b, Alg. 1b] to minimize MMD to $\mathbb{P}$)

Input: kernel $\boldsymbol{k}_{\mathbb{P}}$ with zero-mean under $\mathbb{P}$, input points $\mathcal{S}_n = (x_i)_{i\in[n]}$, candidate coreset indices $(\mathtt{I}^{(\ell)})_{\ell\in[L]}$
$m \leftarrow \left|\mathtt{I}^{(0)}\right|$ ▷ *all candidate coresets are of the same size*
$\mathtt{I} \leftarrow \mathtt{I}^{(\ell^*)}$ for $\ell^* \in \arg\min_{\ell\in[L]} \mathrm{MMD}_{\boldsymbol{k}_{\mathbb{P}}}(\mathcal{S}_n[\mathtt{I}^{(\ell)}], \mathbb{P})$ ▷ *select the best* KT-SPLIT *coreset*
$\mathtt{I}_{\mathrm{ST}} \leftarrow$ index sequence of SteinThinning$(\boldsymbol{k}_{\mathbb{P}}, \mathcal{S}_n, m)$ ▷ *add Stein thinning baseline*
$\mathtt{C} = \{\mathtt{I}, \mathtt{I}_{\mathrm{ST}}\}$ ▷ *shortlisted candidates*
for $\mathtt{I} \in \mathtt{C}$ **do**
  $g \leftarrow \mathbf{0} \in \mathbb{R}^n$ ▷ *maintain sufficient statistics* $g = \sum_{j\in[m]} \boldsymbol{k}_{\mathbb{P}}(x_{\mathtt{I}_j}, \mathcal{S}_n)$
  $\mathtt{Kdiag} \leftarrow (\boldsymbol{k}_{\mathbb{P}}(x_i, x_i))_{i\in[n]}$
  for $j = 1$ **to** m **do**
    $g \leftarrow g + \boldsymbol{k}_{\mathbb{P}}(x_{\mathtt{I}_j}, \mathcal{S}_n)$
  end for
  for $j = 1$ **to** m **do**
    $\Delta = 2(g - \boldsymbol{k}_{\mathbb{P}}(x_{\mathtt{I}_j}, \mathcal{S}_n)) + \mathtt{Kdiag}$ ▷ *this is the change in* $\mathrm{MMD}^2_{\boldsymbol{k}_{\mathbb{P}}}(\mathcal{S}_n[\mathtt{I}], \mathbb{P})$ *if we were to replace* $\mathtt{I}_j$
    $k \leftarrow \arg\min_{i\in[n]} \Delta_i$
    $g = g - \boldsymbol{k}_{\mathbb{P}}(x_{\mathtt{I}_j}, \mathcal{S}_n) + \boldsymbol{k}_{\mathbb{P}}(x_k, \mathcal{S}_n)$
    $\mathtt{I}_j \leftarrow k$
  end for
end for
Return: $\mathtt{I} = \arg\min_{\mathtt{I}\in\mathtt{C}} \mathrm{MMD}_{\boldsymbol{k}_{\mathbb{P}}}(\mathcal{S}_n[\mathtt{I}], \mathbb{P})$

Our KernelThinning implementation is detailed in Alg. 3.D.2. Since we are able to directly compute $\mathrm{MMD}_{\boldsymbol{k}_{\mathbb{P}}}(\mathbb{S}^w_n, \mathbb{P})$, we use KT-Swap (Alg. 3.D.3) in place of the standard KT-SWAP subroutine [DM22b, Algorithm 1b] to choose candidate points to swap in so as to greedily minimize $\mathrm{MMD}_{\boldsymbol{k}_{\mathbb{P}}}(\mathbb{S}^w_n, \mathbb{P})$. To facilitate our subsequent SKT analysis, we restate the guarantees of KT-SPLIT [DM22b, Theorem 2] in the sub-Gaussian format of [SDM22, Definition 3].

Lemma 3.D.1 (Sub-Gaussian guarantee for KT-SPLIT)**.** *Let $\mathcal{S}_n$ be a sequence of n points and $\boldsymbol{k}$ a kernel. For any $\delta \in (0,1)$ and $m \in \mathbb{N}$ such that $\log_2 \frac{n}{m} \in \mathbb{N}$, consider the* KT-SPLIT *algorithm [DM22b, Algorithm 1a] with $\boldsymbol{k}_{\mathrm{split}} = \boldsymbol{k}$, thinning parameter $\mathfrak{t} = \log_2 \frac{n}{m}$, and $\delta_i = \frac{\delta}{n}$ to compress $\mathcal{S}_n$ to $2^{\mathfrak{t}}$ coresets $\{\mathcal{S}^{(i)}_{\mathrm{out}}\}_{i\in[2^{\mathfrak{t}}]}$ where each $\mathcal{S}^{(i)}_{\mathrm{out}}$ has m points. Denote the signed measure $\phi^{(i)} \triangleq \frac{1}{n}\sum_{x\in\mathcal{S}_n} \delta_x - \frac{2^{\mathfrak{t}}}{n}\sum_{x\in\mathcal{S}^{(i)}_{\mathrm{out}}} \delta_x$. Then for each $i \in [2^{\mathfrak{t}}]$, on an event $\mathcal{E}^{(i)}_{\mathrm{equi}}$ with $\mathbb{P}(\mathcal{E}^{(i)}_{\mathrm{equi}}) \geq 1 - \frac{\delta}{2}$, $\phi^{(i)} = \tilde{\phi}^{(i)}$ for a random signed measure $\tilde{\phi}^{(i)}$[5] such that, for*

[5] This is the signed measure returned by repeated applications of self-balancing Hilbert walk (SBHW)

any $\delta' \in (0,1)$,

$$\Pr\left(\left\|\tilde{\phi}^{(i)}\boldsymbol{k}\right\|_{\mathcal{H}_{\boldsymbol{k}}} \geq a_{n,m} + v_{n,m}\sqrt{\log(\tfrac{1}{\delta'})}\right) \leq \delta',$$

where

$$a_{n,m} \triangleq \tfrac{1}{m}\left(2 + \sqrt{\tfrac{8}{3}\|\boldsymbol{k}\|_n \log(\tfrac{6(\log_2 \frac{n}{m})m}{\delta}) \log(4\mathcal{N}_{\boldsymbol{k}}(\mathcal{B}_2(R_n), m^{-1}))}\right),$$
$$v_{n,m} \triangleq \tfrac{1}{m}\sqrt{\tfrac{8}{3}\|\boldsymbol{k}\|_n \log(\tfrac{6(\log_2 \frac{n}{m})m}{\delta})}.$$

Proof of Lem. 3.D.1. Fix $i \in [2^{\mathfrak{t}}]$, $\delta \in (0,1)$ and $n, m \in \mathbb{N}$ such that $\mathfrak{t} = \log_2 \frac{n}{m} \in \mathbb{N}$. Define $\phi \triangleq \phi^{(i)}$. By the proof of Dwivedi and Mackey [DM22b, Thms. 1 and 2], there exists an event $\mathcal{E}_{\text{equi}}$ with $\Pr(\mathcal{E}_{\text{equi}}^c) \leq \frac{\delta}{2}$ such that, on this event, $\phi = \tilde{\phi}$ where $\tilde{\phi}$ is a signed measure such that, for any $\delta' \in (0,1)$, with probability at least $1 - \delta'$,

$$\left\|\tilde{\phi}\boldsymbol{k}\right\|_{\mathcal{H}_{\boldsymbol{k}}} \leq \inf_{\epsilon \in (0,1), A : \mathcal{S}_n \subset A} 2\epsilon + \tfrac{2^{\mathfrak{t}}}{n}\sqrt{\tfrac{8}{3}\|\boldsymbol{k}\|_n \log(\tfrac{6\mathfrak{t}n}{2^{\mathfrak{t}}\delta})\left[\log\tfrac{4}{\delta'} + \log \mathcal{N}_{\boldsymbol{k}}(A, \epsilon)\right]}.$$

Note that on $\mathcal{E}_{\text{equi}}$, $\left\|\tilde{\phi}\boldsymbol{k}\right\|_{\mathcal{H}_{\boldsymbol{k}}} = \|\phi\boldsymbol{k}\|_{\mathcal{H}_{\boldsymbol{k}}}$. We choose $A = \mathcal{B}_2(R_n)$ and $\epsilon = \frac{2^{\mathfrak{t}}}{n} = m^{-1}$, so that, with probability at least $1 - \delta'$, using the fact that $\sqrt{a+b} \leq \sqrt{a} + \sqrt{b}$ for $a, b \geq 0$,

$$\begin{aligned}
\left\|\tilde{\phi}\boldsymbol{k}\right\|_{\mathcal{H}_{\boldsymbol{k}}} &\leq \tfrac{2^{\mathfrak{t}+1}}{n} + \tfrac{2^{\mathfrak{t}}}{n}\sqrt{\tfrac{8}{3}\|\boldsymbol{k}\|_n \log(\tfrac{6\mathfrak{t}n}{2^{\mathfrak{t}}\delta})\left[\log\tfrac{4}{\delta'} + \log \mathcal{N}_{\boldsymbol{k}}(\mathcal{B}_2(R_n), m^{-1})\right]} \\
&\leq \tfrac{2^{\mathfrak{t}+1}}{n} + \tfrac{2^{\mathfrak{t}}}{n}\sqrt{\tfrac{8}{3}\|\boldsymbol{k}\|_n \log(\tfrac{6\mathfrak{t}n}{2^{\mathfrak{t}}\delta})}\left[\sqrt{\log\tfrac{1}{\delta'}} + \sqrt{\log 4\mathcal{N}_{\boldsymbol{k}}(\mathcal{B}_2(R_n), m^{-1})}\right] \\
&\leq a_{n,m} + v_{n,m}\sqrt{\log(\tfrac{1}{\delta'})},
\end{aligned} \tag{3.40}$$

for $a_{n,m}$, $v_{n,m}$ in Lem. 3.D.1. □

Corollary 3.D.1 (MMD guarantee for KT-SPLIT)**.** *Let $\mathcal{S}_\infty$ be an infinite sequence of points in $\mathbb{R}^d$ and $\boldsymbol{k}$ a kernel. For any $\delta \in (0,1)$ and $n, m \in \mathbb{N}$ such that $\log_2 \frac{n}{m} \in \mathbb{N}$, consider the KT-SPLIT algorithm [DM22b, Algorithm 1a] with parameters $\boldsymbol{k}_{\text{split}} = \boldsymbol{k}$ and $\delta_i = \frac{\delta}{n}$ to compress $\mathcal{S}_n$ to $2^{\mathfrak{t}}$ coresets $\{\mathcal{S}_{\text{out}}^{(i)}\}_{i \in [2^{\mathfrak{t}}]}$ where $\mathfrak{t} = \log_2 \frac{n}{m}$, each with m points. Then for any $i \in [2^{\mathfrak{t}}]$, with probability at least $1 - \delta$,*

$$\text{MMD}_{\boldsymbol{k}}(\mathbb{S}_n, \mathbb{S}_{\text{out}}^{(i)}) \leq \tfrac{1}{m}\left(2 + \sqrt{\tfrac{8}{3}\|\boldsymbol{k}\|_n \log(\tfrac{6(\log_2 \frac{n}{m})m}{\delta})\left(\log \mathcal{N}_{\boldsymbol{k}}(\mathcal{B}_2(R_n), m^{-1}) + \log\tfrac{8}{\delta}\right)}\right). \tag{3.41}$$

[DM21, Algorithm 3]. Although SBHW returns an element of $\mathcal{H}_{\boldsymbol{k}}$, by tracing the algorithm, the returned output is equivalent to a signed measure via the correspondence $\sum_{i\in[n]} c_i \boldsymbol{k}(x_i, \cdot) \Leftrightarrow \sum_{i\in[n]} c_i \delta_{x_i}$. The usage of signed measures is consistent with Shetty, Dwivedi, and Mackey [SDM22].

Proof of Cor. 3.D.1. Fix $i \in [2^t]$. By taking $\delta' = \frac{\delta}{2}$ in (3.40), we obtain (3.41). This occurs with probability

$$\begin{aligned}
&\Pr(\mathrm{MMD}_{\boldsymbol{k}}(\mathbb{S}_n, \mathbb{S}_{\mathrm{out}}^{(i)}) < a_{n,m} + v_{n,m}\sqrt{\log(\tfrac{1}{\delta'})}) \\
=&1 - \Pr\left(\mathrm{MMD}_{\boldsymbol{k}}(\mathbb{S}_n, \mathbb{S}_{\mathrm{out}}^{(i)}) \geq a_{n,m} + v_{n,m}\sqrt{\log(\tfrac{1}{\delta'})}\right) \\
\geq&1 - \Pr\left(\mathcal{E}_{\mathrm{equi}}^{(i)}, \mathrm{MMD}_{\boldsymbol{k}}(\mathbb{S}_n, \mathbb{S}_{\mathrm{out}}^{(i)}) \geq a_{n,m} + v_{n,m}\sqrt{\log(\tfrac{1}{\delta'})}\right) - \Pr\left(\mathcal{E}_{\mathrm{equi}}^{(i)\,c}\right) \\
\geq&1 - \Pr\left(\left\|\tilde{\phi}^{(i)}\boldsymbol{k}\right\|_{\mathcal{H}_{\boldsymbol{k}}} \geq a_{n,m} + v_{n,m}\sqrt{\log(\tfrac{1}{\delta'})}\right) - \Pr(\mathcal{E}_{\mathrm{equi}}^{(i)\,c}) \\
\geq&1 - \tfrac{\delta}{2} - \tfrac{\delta}{2} = 1 - \delta.
\end{aligned}$$

□

♢ 3.D.3 Proof of Thm. 3.3: MMD guarantee for SKT

Thm. 3.3 will follows directly from Assum. $(\boldsymbol{\alpha}, \boldsymbol{\beta})$-kernel and the following statement for a generic covering number.

Theorem 3.D.1. *Let $\boldsymbol{k}_{\mathbb{P}}$ be a kernel satisfying Assum. 3.1. Let $\mathcal{S}_\infty$ be an infinite sequence of points. Then for a prefix sequence $\mathcal{S}_n$ of n points, $m \in [n]$, and $n' \triangleq m2^{\lceil \log_2 \frac{n}{m} \rceil}$, SKT outputs w_{SKT} in $O(n^2 d_{\boldsymbol{k}_{\mathbb{P}}})$ time that satisfies, with probability at least $1-\delta$,*

$$\Delta\mathrm{MMD}_{\boldsymbol{k}_{\mathbb{P}}}(w_{\mathrm{SKT}}) \leq \sqrt{\left(\tfrac{1+\log n'}{n'}\right)\|\boldsymbol{k}_{\mathbb{P}}\|_n} + \frac{1}{m}\left(2 + \sqrt{\tfrac{8}{3}\|\boldsymbol{k}\|_n \log(\tfrac{6(\log_2 \frac{n'}{m})m}{\delta})\left(\log \mathcal{N}_{\boldsymbol{k}}(\mathcal{B}_2(R_n), m^{-1}) + \log\tfrac{8}{\delta}\right)}\right).$$

Proof of Thm. 3.D.1. The runtime of SKT comes from the fact that all of SteinThinning (with output size n), KT-SPLIT, and KT-Swap take $O(d_{\boldsymbol{k}_{\mathbb{P}}} n^2)$ time.

By Riabiz et al. [Ria+22, Theorem 1], SteinThinning (which is a deterministic algorithm) from n points to n' points has the following guarantee

$$\mathrm{MMD}^2_{\boldsymbol{k}_{\mathbb{P}}}(\mathbb{S}_n^{w^\dagger}, \mathbb{P}) \leq \mathrm{MMD}^2_{\boldsymbol{k}_{\mathbb{P}}}(\mathbb{S}_n^{w_{\mathrm{OPT}}}, \mathbb{P}) + \left(\tfrac{1+\log n'}{n'}\right)\|\boldsymbol{k}_{\mathbb{P}}\|_n,$$

where we denote the output weight of SteinThinning as $w^\dagger$. Using $\sqrt{a+b} \leq \sqrt{a} + \sqrt{b}$ for $a, b \geq 0$, we have

$$\mathrm{MMD}_{\boldsymbol{k}_{\mathbb{P}}}(\mathbb{S}_n^{w^\dagger}, \mathbb{P}) \leq \mathrm{MMD}_{\boldsymbol{k}_{\mathbb{P}}}(\mathbb{S}_n^{w_{\mathrm{OPT}}}, \mathbb{P}) + \sqrt{\left(\tfrac{1+\log n'}{n'}\right)\|\boldsymbol{k}_{\mathbb{P}}\|_n}.$$

Fix $\delta \in (0,1)$. By Cor. 3.D.1 with $\boldsymbol{k} = \boldsymbol{k}_{\mathbb{P}}$ and $\mathfrak{t} = \log_2 \frac{n'}{m}$, with probability at least $1-\delta$, we have, for any $i \in [2^{\mathfrak{t}}]$,

$$\mathrm{MMD}_{\boldsymbol{k}_{\mathbb{P}}}(\mathbb{S}_n^{w^\dagger}, \mathbb{S}_{\mathrm{out}}^{(i)}) \leq \frac{1}{m}\left(2 + \sqrt{\tfrac{8}{3}\|\boldsymbol{k}\|_n \log(\tfrac{6(\log_2 \frac{n'}{m})m}{\delta})\left(\log \mathcal{N}_{\boldsymbol{k}}(\mathcal{B}_2(R_n), m^{-1}) + \log \tfrac{8}{\delta}\right)}\right),$$

where $\mathbb{S}_{\mathrm{out}}^{(i)}$ is the i-th coreset output by KT-SPLIT. Since KT-Swap can only decrease the MMD to $\mathbb{P}$, we have, by the triangle inequality of $\mathrm{MMD}_{\boldsymbol{k}_{\mathbb{P}}}$,

$$\mathrm{MMD}_{\boldsymbol{k}_{\mathbb{P}}}(\mathbb{S}_n^{w_{\mathrm{SKT}}}, \mathbb{P}) \leq \mathrm{MMD}_{\boldsymbol{k}_{\mathbb{P}}}(\mathbb{S}_{\mathrm{out}}^{(1)}, \mathbb{P}) \leq \mathrm{MMD}_{\boldsymbol{k}_{\mathbb{P}}}(\mathbb{S}_{\mathrm{out}}^{(1)}, \mathbb{S}_n^{w^\dagger}) + \mathrm{MMD}_{\boldsymbol{k}_{\mathbb{P}}}(\mathbb{S}_n^{w^\dagger}, \mathbb{P}),$$

which gives the desired bound. □

Thm. 3.3 now follows from Thm. 3.D.1, the kernel growth definitions in Assum. ($\boldsymbol{\alpha}, \boldsymbol{\beta}$)-kernel, $n \leq n' \leq 2n$, and that $\log_2(\frac{n'}{m})m \leq n'$. □

■ 3.E Resampling of Simplex Weights

Integral to many of our algorithms is a resampling procedure that turns a simplex-weighted point set of size n into an equal-weighted point set of size m while incurring at most $O(1/\sqrt{m})$ MMD error. The motivation for wanting an equal-weighted point set is two-fold: First, in LSKT, we need to provide an equal-weighted point set to KT-Compress++, but the output of LD is a simplex weight. Secondly, we can exploit the fact that non-zero weights are bounded away from zero in equal-weighted point sets to provide a tighter analysis of WeightedRPCholesky. While i.i.d. resampling also achieves the $O(1/\sqrt{m})$ goal, we choose Resample (Alg. 3.E.3), a stratified residual resampling algorithm [DC05, Sec. 3.2, 3.3]. In this section, we derive an MMD bound for Resample and show that it is better in expectation than using i.i.d. resampling or residual resampling alone.

Let D_w^{inv} be the inverse of the cumulative distribution function of the multinomial distribution with weight w, i.e.,

$$D_w^{\mathrm{inv}}(u) \triangleq \min\left\{i \in [n] : u \leq \textstyle\sum_{j=1}^{i} w_j\right\}.$$

Proposition 3.E.1 (MMD guarantee of resampling algorithms)**.** *Consider any kernel $\boldsymbol{k}$, points $\mathcal{S}_n = (x_1, \ldots, x_n) \subset \mathbb{R}^d$, and a weight vector $w \in \Delta_{n-1}$.*

(a) Using the notation from Alg. 3.E.1, let X, X' be independent random variables with law $\mathbb{S}_n^w$. Then, the output weight vector $w^{\mathrm{i.i.d.}} \triangleq w'$ of Alg. 3.E.1 satisfies

$$\mathbb{E}[\mathrm{MMD}_{\boldsymbol{k}}^2(\mathbb{S}_n^{w^{\mathrm{i.i.d.}}}, \mathbb{S}_n^w)] = \tfrac{\mathbb{E}\boldsymbol{k}(X,X) - \mathbb{E}\boldsymbol{k}(X,X')}{m}. \tag{3.42}$$

Algorithm 3.E.1 i.i.d. resampling

Input: Weights $w \in \Delta_{n-1}$, output size m
$w' \leftarrow \mathbf{0} \in \mathbb{R}^n$
for $j = 1$ **to** m **do**
  Draw $U_j \sim \mathrm{Uniform}([0,1))$
  $\mathtt{I}_j \leftarrow D_w^{\mathrm{inv}}(U_j)$
  $w'_{\mathtt{I}_j} \leftarrow w'_{\mathtt{I}_j} + \frac{1}{m}$
end for
Return: $w' \in \Delta_{n-1} \cap (\frac{\mathbb{N}_0}{m})^n$

Algorithm 3.E.2 Residual resampling

Input: Weights $w \in \Delta_{n-1}$, output size m
$w'_i \leftarrow \frac{\lfloor mw_i \rfloor}{m}$, $\forall i \in [n]$
$r \leftarrow m - \sum_{i\in[n]} \lfloor mw_i \rfloor \in \mathbb{N}$
$\eta_i \leftarrow \frac{mw_i - \lfloor mw_i \rfloor}{r}$, $\forall i \in [n]$
for $j = 1$ **to** r **do**
  Draw $U_j \sim \mathrm{Uniform}([0,1))$
  $\mathtt{I}_j \leftarrow D_\eta^{\mathrm{inv}}(U_j)$
  $w'_{\mathtt{I}_j} \leftarrow w'_{\mathtt{I}_j} + \frac{1}{m}$
end for
Return: $w' \in \Delta_{n-1} \cap (\frac{\mathbb{N}_0}{m})^n$

(b) Using the notation from Alg. 3.E.2, let R, R' be independent random variables with law $\mathbb{S}_n^\eta$. Then, the output weight vector $w^{\mathrm{resid}} \triangleq w'$ of Alg. 3.E.2 satisfies

$$\mathbb{E}[\mathrm{MMD}_{\boldsymbol{k}}^2(\mathbb{S}_n^{w^{\mathrm{resid}}}, \mathbb{S}_n^w)] = \tfrac{r(\mathbb{E}\boldsymbol{k}(R,R) - \mathbb{E}\boldsymbol{k}(R,R'))}{m^2}. \tag{3.43}$$

(c) Using the notation from Alg. 3.E.3, let $R_j \triangleq x_{\mathtt{I}_j}$ and R'_j be an independent copy of R_j. Let R be an independent random variable with law $\mathbb{S}_n^\eta$. Then, the output weight vector $w^{\mathrm{sr}} \triangleq w'$ of Alg. 3.E.3 satisfies

$$\mathbb{E}[\mathrm{MMD}_{\boldsymbol{k}}^2(\mathbb{S}_n^{w^{\mathrm{sr}}}, \mathbb{S}_n^w)] = \tfrac{r\mathbb{E}\boldsymbol{k}(R,R) - \sum_{j\in[r]} \mathbb{E}\boldsymbol{k}(R_j, R'_j)}{m^2}. \tag{3.44}$$

Proof of Prop. 3.E.1(a). Let $X_i \triangleq x_{\mathtt{I}_i}$. As random signed measures, we have

$$\mathbb{S}_n^{w'} - \mathbb{S}_n^w = \tfrac{1}{m} \textstyle\sum_{i\in[m]} \delta_{X_i} - \sum_{i\in[n]} w_i \delta_{x_i}.$$

Algorithm 3.E.3 Stratified residual resampling (Resample)

Input: Weights $w \in \Delta_{n-1}$, output size m
$w'_i \leftarrow \frac{\lfloor mw_i \rfloor}{m}$, $\forall i \in [n]$
$r \leftarrow m - \sum_{i\in[n]} \lfloor mw_i \rfloor \in \mathbb{N}$
$\eta_i \leftarrow \frac{mw_i - \lfloor mw_i \rfloor}{r}$, $\forall i \in [n]$
for $j = 1$ **to** r **do**
  Draw $U_j \sim \mathrm{Uniform}([\frac{j}{r}, \frac{j+1}{r}))$
  $\mathtt{I}_j \leftarrow D_\eta^{\mathrm{inv}}(U_j)$
  $w'_{\mathtt{I}_j} \leftarrow w'_{\mathtt{I}_j} + \frac{1}{m}$
end for
Return: $w' \in \Delta_{n-1} \cap (\frac{\mathbb{N}_0}{m})^n$

Hence

$$\begin{aligned}
&\mathrm{MMD}^2_{\boldsymbol{k}}(\mathbb{S}_n^{w'}, \mathbb{S}_n^{w}) \\
&= ((\mathbb{S}_n^{w'} - \mathbb{S}_n^{w}) \times (\mathbb{S}_n^{w'} - \mathbb{S}_n^{w}))\boldsymbol{k} \\
&= \tfrac{1}{m^2} \textstyle\sum_{i,i'\in[m]} \boldsymbol{k}(X_i, X_{i'}) - \tfrac{2}{m} \sum_{i\in[m],i'\in[n]} w_{i'} \boldsymbol{k}(X_i, x_{i'}) + \sum_{i,i'\in[n]} w_i w_{i'} \boldsymbol{k}(x_i, x_{i'}).
\end{aligned}$$

Since each X_i is distributed to $\mathbb{S}_n^w$ and X_i and $X_{i'}$ are independent for $i \neq i'$, taking expectation, we have

$$\mathbb{E}[\mathrm{MMD}^2_{\boldsymbol{k}}(\mathbb{S}_n^{w'}, \mathbb{S}_n^{w})] = \tfrac{1}{m}\mathbb{E}\boldsymbol{k}(X, X) + \tfrac{m-1}{m}\mathbb{E}\boldsymbol{k}(X, X') - 2\mathbb{E}\boldsymbol{k}(X, X') + \mathbb{E}\boldsymbol{k}(X, X').$$

This gives the bound (3.42). □

Proof of Prop. 3.E.1(b). Let $R_j \triangleq x_{\mathtt{I}_j}$. As random signed measures, we have

$$\begin{aligned}
\mathbb{S}_n^{w'} - \mathbb{S}_n^{w} &= \left(\textstyle\sum_{i\in[n]} \frac{\lfloor mw_i \rfloor}{m} \delta_{x_i} + \frac{1}{m} \sum_{j\in[r]} \delta_{R_j}\right) - \sum_{i\in[n]} w_i \delta_{x_i} \\
&= \tfrac{1}{m} \textstyle\sum_{j\in[r]} \delta_{R_j} - \sum_{i\in[n]} \left(w_i - \frac{\lfloor mw_i \rfloor}{m}\right) \delta_{x_i} \\
&= \tfrac{1}{m} \textstyle\sum_{j\in[r]} \delta_{R_j} - \frac{r}{m} \sum_{i\in[n]} \eta_i \delta_{x_i}.
\end{aligned}$$

Hence

$$\begin{aligned}
&\mathrm{MMD}^2_{\boldsymbol{k}}(\mathbb{S}_n^{w'}, \mathbb{S}_n^{w}) \\
&= ((\mathbb{S}_n^{w'} - \mathbb{S}_n^{w}) \times (\mathbb{S}_n^{w'} - \mathbb{S}_n^{w}))\boldsymbol{k} \\
&= \tfrac{1}{m^2} \textstyle\sum_{j,j'\in[r]} \boldsymbol{k}(R_j, R_{j'}) - \frac{2r}{m^2} \sum_{j\in[r],i\in[n]} \eta_i \boldsymbol{k}(R_j, x_i) + \frac{r^2}{m^2} \sum_{i,i'\in[n]} \eta_i \eta_j \boldsymbol{k}(x_i, x_j). \qquad (3.45)
\end{aligned}$$

Since each R_j is distributed to $\mathbb{S}_n^\eta$ and R_j and $R_{j'}$ are independent for $j \neq j'$, taking expectation, we have

$$\mathbb{E}[\mathrm{MMD}^2_{\boldsymbol{k}}(\mathbb{S}_n^{w'}, \mathbb{S}_n^{w})] = \tfrac{r}{m^2}\mathbb{E}\boldsymbol{k}(R, R) + \tfrac{r(r-1)}{m^2}\mathbb{E}\boldsymbol{k}(R, R') - \tfrac{2r^2}{m^2}\mathbb{E}\boldsymbol{k}(R, R') + \tfrac{r^2}{m^2}\mathbb{E}\boldsymbol{k}(R, R').$$

This gives the bound (3.43). □

Proof of Prop. 3.E.1(c). We repeat the same steps from the previous part of the proof to get (3.45). In the case of (c), R_j's are not identically distributed so the analysis is different. Let R' be an independent copy of R. Taking expectation of (3.45), we have

$$\begin{aligned}&m^2\mathbb{E}[\mathrm{MMD}^2_{\boldsymbol{k}}(\mathbb{S}^{w'}_n, \mathbb{S}^{w}_n)]\\ &= \textstyle\sum_{j\in[r]} \mathbb{E}\boldsymbol{k}(R_j, R_j) + \sum_{j\in[r]}\sum_{j'\in[r]\setminus\{j\}} \mathbb{E}\boldsymbol{k}(R_j, R_{j'}) - 2r\sum_{j\in[r]} \mathbb{E}\boldsymbol{k}(R_j, R) + r^2\mathbb{E}\boldsymbol{k}(R, R').\end{aligned}$$

Note

$$\begin{aligned}\textstyle\sum_{j\in[r]} \mathbb{E}\boldsymbol{k}(R_j, R_j) &= \textstyle\sum_{j\in[r]} r\int_{[\frac{j}{r}, \frac{j+1}{r})} \boldsymbol{k}(x_{D^{\mathrm{inv}}_\eta(u)}, x_{D^{\mathrm{inv}}_\eta(u)})\mathrm{d}u\\ &= r\textstyle\int_0^1 \boldsymbol{k}(x_{D^{\mathrm{inv}}_\eta(u)}, x_{D^{\mathrm{inv}}_\eta(u)})\mathrm{d}u = r\mathbb{E}\boldsymbol{k}(R, R),\end{aligned}$$

where we used the fact that $x_{D^{\mathrm{inv}}_\eta(U)} \overset{D}{=} R$ for $U \sim \mathrm{Uniform}([0,1])$. Similarly, we deduce

$$\begin{aligned}\textstyle\sum_{j\in[r]}\sum_{j'\in[r]\setminus\{j\}} \mathbb{E}\boldsymbol{k}(R_j, R_{j'}) &= \textstyle\sum_{j\in[r]} \left(\sum_{j'\in[r]} \mathbb{E}\boldsymbol{k}(R_j, R'_{j'}) - \mathbb{E}\boldsymbol{k}(R_j, R'_j)\right)\\ &= \textstyle\sum_{j\in[r]} (r\mathbb{E}\boldsymbol{k}(R_j, R') - \mathbb{E}\boldsymbol{k}(R_j, R'_j))\\ &= r^2\mathbb{E}\boldsymbol{k}(R, R') - \textstyle\sum_{j\in[r]} \mathbb{E}\boldsymbol{k}(R_j, R'_j),\end{aligned}$$

and also

$$\textstyle\sum_{j\in[r]} \mathbb{E}\boldsymbol{k}(R_j, R) = r\mathbb{E}\boldsymbol{k}(R, R').$$

Combining terms, we get

$$\begin{aligned}&m^2\mathbb{E}\,\mathrm{MMD}^2_{\boldsymbol{k}}(\mathbb{S}^{w'}_n, \mathbb{S}^{w}_n)\\ &= r\mathbb{E}\boldsymbol{k}(R, R) + r^2\mathbb{E}\boldsymbol{k}(R, R') - \textstyle\sum_{j\in[r]} \mathbb{E}\boldsymbol{k}(R_j, R'_j) - 2r^2\mathbb{E}\boldsymbol{k}(R, R') + r^2\mathbb{E}\boldsymbol{k}(R, R')\\ &= r\mathbb{E}\boldsymbol{k}(R, R) - \textstyle\sum_{j\in[r]} \mathbb{E}\boldsymbol{k}(R_j, R'_j),\end{aligned}$$

which yields the desired bound (3.44). □

The next proposition shows that stratifying the residuals always improves upon using i.i.d. sampling or residual resampling alone. We need the following convexity lemma.

Lemma 3.E.1 (Convexity of squared MMD)**.** *Let $\boldsymbol{k}$ be a kernel. Let $\mathcal{S}_n = (x_1, \ldots, x_n)$ be an arbitrary set of points. The function $E_{\boldsymbol{k}} : \mathbb{R}^n \to \mathbb{R}$ defined by*

$$E_{\boldsymbol{k}}(w) \triangleq \|\mathbb{S}^w_n \boldsymbol{k}\|^2_{\mathcal{H}_{\boldsymbol{k}}} = \textstyle\sum_{i,j\in[n]} w_i w_j \boldsymbol{k}(x_i, x_j)$$

is convex on $\mathbb{R}^n$.

Proof of Lem. 3.E.1. Since $\boldsymbol{k}$ is a kernel, the Hessian $\mathrm{H}\, E_{\boldsymbol{k}} = 2\boldsymbol{k}(\mathcal{S}_n, \mathcal{S}_n)$ is PSD, and hence $E_{\boldsymbol{k}}$ is convex. □

Proposition 3.E.2 (Stratified residual resampling improves MMD)**.** *Under the assumptions of Prop. 3.E.1, we have*

$$\mathbb{E}[\mathrm{MMD}^2_{\boldsymbol{k}}(\mathbb{S}_n^{w^{\text{i.i.d.}}}, \mathbb{S}_n^w)] \geq \mathbb{E}[\mathrm{MMD}^2_{\boldsymbol{k}}(\mathbb{S}_n^{w^{\text{resid}}}, \mathbb{S}_n^w)] \geq \mathbb{E}[\mathrm{MMD}^2_{\boldsymbol{k}}(\mathbb{S}_n^{w^{\text{sr}}}, \mathbb{S}_n^w)].$$

Proof of Prop. 3.E.2. Let $K \triangleq \boldsymbol{k}(\mathcal{S}_n, \mathcal{S}_n)$. To show the first inequality, note that since $\eta = \frac{mw - \lfloor mw \rfloor}{r}$, by Prop. 3.E.1,

$$\begin{aligned}
\mathbb{E}[\mathrm{MMD}^2_{\boldsymbol{k}}(\mathbb{S}_n^{w^{\text{resid}}}, \mathbb{S}_n^w)] &= \tfrac{r(\mathbb{E}\boldsymbol{k}(R,R) - \mathbb{E}\boldsymbol{k}(R,R'))}{m^2} \\
&= \tfrac{r(\sum_{i\in[n]} K_{ii}\eta_i - \sum_{i,j\in[n]} K_{ij}\eta_i\eta_j)}{m^2} \\
&= \tfrac{1}{m}\left(\sum_{i\in[n]} K_{ii}\left(w_i - \tfrac{\lfloor mw_i \rfloor}{m}\right) - \tfrac{m}{r}\left(w - \tfrac{\lfloor mw \rfloor}{m}\right)^\top K \left(w - \tfrac{\lfloor mw \rfloor}{m}\right)\right).
\end{aligned}$$

Hence

$$\begin{aligned}
&\mathbb{E}[\mathrm{MMD}^2_{\boldsymbol{k}}(\mathbb{S}_n^{w^{\text{i.i.d.}}}, \mathbb{S}_n^w)] - \mathbb{E}[\mathrm{MMD}^2_{\boldsymbol{k}}(\mathbb{S}_n^{w^{\text{resid}}}, \mathbb{S}_n^w)] \\
=&\tfrac{1}{m}\left(\sum_{i\in[n]} K_{ii}\tfrac{\lfloor mw_i \rfloor}{m} + \tfrac{m}{r}\left(w - \tfrac{\lfloor mw \rfloor}{m}\right)^\top K \left(w - \tfrac{\lfloor mw \rfloor}{m}\right) - w^\top K w\right) \\
=&\tfrac{1}{m}\left((1-\theta)\sum_{i\in[n]} K_{ii}\xi_i + \theta\eta^\top K\eta - w^\top K w\right),
\end{aligned}$$

where we let $\xi \triangleq \frac{m}{m-r}\frac{\lfloor mw \rfloor}{m}$ and $\theta \triangleq \frac{r}{m}$. Note that $w = \theta\eta + (1-\theta)\xi$. By Lem. 3.E.1 and Jensen's inequality, we have

$$\begin{aligned}
w^\top K w = E_{\boldsymbol{k}}(w) &\leq \theta E_{\boldsymbol{k}}(\eta) + (1-\theta)E_{\boldsymbol{k}}(\xi) \\
&= \theta\eta^\top K\eta + (1-\theta)\xi^\top K\xi \leq \theta\eta^\top K\eta + (1-\theta)\textstyle\sum_{i\in[n]} K_{ii}\xi_i,
\end{aligned}$$

where the last inequality follows from Prop. 3.E.1(a) with $w = \xi$ and the fact that MMD is nonnegative. Hence we have shown

$$\mathbb{E}[\mathrm{MMD}^2_{\boldsymbol{k}}(\mathbb{S}_n^{w^{\text{i.i.d.}}}, \mathbb{S}_n^w)] - \mathbb{E}[\mathrm{MMD}^2_{\boldsymbol{k}}(\mathbb{S}_n^{w^{\text{resid}}}, \mathbb{S}_n^w)] \geq 0,$$

as desired.

For the second inequality, by Prop. 3.E.1, we compute

$$\mathbb{E}[\mathrm{MMD}^2_{\boldsymbol{k}}(\mathbb{S}_n^{w^{\text{resid}}}, \mathbb{S}_n^w)] - \mathbb{E}[\mathrm{MMD}^2_{\boldsymbol{k}}(\mathbb{S}_n^{w^{\text{sr}}}, \mathbb{S}_n^w)] = \tfrac{r}{m^2}\left(\tfrac{1}{r}\textstyle\sum_{j\in[r]} \mathbb{E}\boldsymbol{k}(R_j, R'_j) - \mathbb{E}\boldsymbol{k}(R, R')\right).$$

Note that

$$\mathbb{E}\boldsymbol{k}(R,R') = \textstyle\int_{[0,1)}\int_{[0,1)} k(x_{D_\eta^{\text{inv}}(u)}, x_{D_\eta^{\text{inv}}(v)})\mathrm{d}u\mathrm{d}v = E_{\boldsymbol{k}}\left((D_\eta^{\text{inv}})_{\#}\mathrm{Uniform}[0,1)\right),$$

where we used $T_{\#}\mu$ to denote the pushforward measure of μ by T. Similarly,

$$\begin{aligned}\tfrac{1}{r}\sum_{j\in[r]}\mathbb{E}\boldsymbol{k}(R_j,R'_j) &= \tfrac{1}{r}\sum_{j\in[r]}\int_{[\frac{j}{r},\frac{j+1}{r})}\int_{[\frac{j}{r},\frac{j+1}{r})}k(x_{D_\eta^{\mathrm{inv}}(u)},x_{D_\eta^{\mathrm{inv}}(v)})\mathrm{d}u\mathrm{d}v\\ &= \tfrac{1}{r}\sum_{j\in[r]}E_{\boldsymbol{k}}\left((D_\eta^{\mathrm{inv}})_{\#}\mathrm{Uniform}\left[\tfrac{j}{r},\tfrac{j+1}{r}\right)\right)\\ &\le E_{\boldsymbol{k}}\left((D_\eta^{\mathrm{inv}})_{\#}\mathrm{Uniform}[0,1)\right)=\mathbb{E}\boldsymbol{k}(R,R'),\end{aligned}$$

where in the last inequality we applied Jensen's inequality since $E_{\boldsymbol{k}}$ is convex by Lem. 3.E.1. Hence we have shown

$$\mathbb{E}[\mathrm{MMD}^2_{\boldsymbol{k}}(\mathbb{S}_n^{w^{\mathrm{resid}}},\mathbb{S}_n^w)]-\mathbb{E}[\mathrm{MMD}^2_{\boldsymbol{k}}(\mathbb{S}_n^{w^{\mathrm{sr}}},\mathbb{S}_n^w)]\ge 0$$

and the proof is complete. □

■ 3.F Accelerated Debiased Compression

In this section, we provide supplementary algorithmic details and deferred analyses for LSKT (Alg. 3.4). In WeightedRPCholesky (Alg. 3.F.1), we provide details for the weighted extension of Chen et al. [Che+22b, Alg. 2.1] that is used extensively in our algorithms. The details of AMD [WAL23, Alg. 14] are provided in Alg. 3.F.2. In Sec. 3.F.1, we give the proof of Thm. 3.4 for the MMD error guarantee of LD (Alg. 3.3). In Sec. 3.F.2, we provide details on KT-Compress++ modified from Compress++ [SDM22] to minimize MMD to $\mathbb{P}$. Finally, Thm. 3.5 is proved in Sec. 3.F.3.

Algorithm 3.F.1 Weighted Randomly Pivoted Cholesky (WeightedRPCholesky) (extension of Chen et al. [Che+22b, Alg. 2.1])

Input: kernel $\boldsymbol{k}$, points $\mathcal{S}_n=(x_i)_{i=1}^n$, simplex weights $w\in\Delta_{n-1}$, rank r
$\tilde{\boldsymbol{k}}(i,j)\triangleq\boldsymbol{k}(x_i,x_j)\sqrt{w_i}\sqrt{w_j}$ ▷ *reweighted kernel matrix function*
$F\leftarrow\mathbf{0}_{n\times r}$, $\mathtt{S}\leftarrow\{\}$, $d\leftarrow(\tilde{\boldsymbol{k}}(i,i))_{i\in[n]}$
for $i=1$ **to** r **do**
  Sample $s\sim d/\sum_{j\in[n]}d_j$
  $\mathtt{S}\leftarrow\mathtt{S}\cup\{s\}$
  $g\leftarrow\tilde{\boldsymbol{k}}(:,s)-F(:,1:i-1)F(s,1:i-1)^\top$
  $F(:,i)\leftarrow g/\sqrt{g_s}$
  $d\leftarrow d-F(:,i)^2$ ▷ *$F(:,i)^2$ denotes a vector with entry-wise squared values of $F(:,i)$*
  $d\leftarrow\max(d,0)$ ▷ *numerical stability fix, helpful in practice*
end for
$F\leftarrow\mathrm{diag}((1/\sqrt{w_i})_{i\in[n]})F$ ▷ *undo weighting; treat $1/\sqrt{w_i}=0$ if $w_i=0$*
Return: $\mathtt{S}\subset[n]$ with $|\mathtt{S}|=r$ and $F\in\mathbb{R}^{n\times r}$

Algorithm 3.F.2 Accelerated Entropic Mirror Descent (AMD) (modification of Wang, Abernethy, and Levy [WAL23, Alg. 14])

Input: kernel matrix $K \in \mathbb{R}^{n\times n}$, number of steps T, initial weight $w_0 \in \Delta_{n-1}$, aggressive flag `AGG`
$\eta \leftarrow \frac{1}{8w_0^\top \operatorname{diag}(K)}$ if `AGG` else $\frac{1}{8\max_{i\in[n]} K_{ii}}$
$v_0 \leftarrow w_0$
for $t = 1$ **to** T **do**
  $\beta_t \leftarrow \frac{2}{t+1}$
  $z_t \leftarrow (1-\beta_t)w_{t-1} + \beta_t v_{t-1}$
  $g \leftarrow 2t\eta K z_t$ ▷ *this is $\gamma_t \nabla f(z_t)$ in Wang, Abernethy, and Levy [WAL23, Alg. 14] for $f(w) = w^\top K w$*
  $v_t \leftarrow v_{t-1} \cdot \exp(-g)$ ▷ *component-wise exponentiation and multiplication*
  $v_t \leftarrow v_t / \|v_t\|_1$ ▷ *$v_t = \arg\min_{w\in\Delta_{n-1}} \langle g, w\rangle + D^\phi_{v_{t-1}}(w)$ for $\phi(w) = \sum_{i\in[n]} w_i \log w_i$*
  $w_t \leftarrow (1-\beta_t)w_{t-1} + \beta_t v_t$
end for
Return: $w_T \in \Delta_{n-1}$

♢ 3.F.1 Proof of Thm. 3.4: Debiasing guarantee for LD

We start with a useful lemma that bounds $w^\top(K - \hat{K})w$ by $\operatorname{tr}(K - \hat{K})$ for any simplex weights w.

Lemma 3.F.1. *For any PSD matrix $A \in \mathbb{R}^{n\times n}$ and $w \in \Delta_{n-1}$, we have*

$$w^\top A w \le \operatorname{tr}(A^w) \le \max_{i\in[n]} A_{ii} \le \lambda_1(A),$$

where $\lambda_1(A)$ denotes the largest eigenvalue of A.

Proof of Lem. 3.F.1. Note that

$$w^\top A w = \sqrt{w}^\top \operatorname{diag}(\sqrt{w}) A \operatorname{diag}(\sqrt{w})\sqrt{w} = \sqrt{w}^\top A^w \sqrt{w}.$$

The condition that $w \in \Delta_{n-1}$ implies $\|\sqrt{w}\|_2 = 1$, so that

$$\sqrt{w}^\top A^w \sqrt{w} \le \lambda_1(A^w) \le \operatorname{tr}(A^w).$$

To see $\operatorname{tr}(A^w) \le \max i \in [n] A_{ii}$, note that $\operatorname{tr}(A^w) = \sum_{i\in[n]} A_{ii} w_i \le \max_{i\in[n]} A_{ii}$ since $w \in \Delta_{n-1}$..

Since $\lambda_1(A) = \sup_{x:\|x\|_2=1} x^\top A x$, if we let $i^* \triangleq \arg\min_{i\in[n]} A_{ii}$, then the simplex weight with 1 on the i^*-th entry has two-norm 1, so we see that $\max_{i\in[n]} A_{ii} \le \lambda_1(A)$. □

Our next lemma bounds the suboptimality of surrogate optimization of a low-rank plus diagonal approximation of K.

Lemma 3.F.2 (Suboptimality of surrogate optimization)**.** *Let $\boldsymbol{k}_{\mathbb{P}}$ be a kernel satisfying Assum. 3.1. Let $\mathcal{S}_n = (x_1, \dots, x_n) \subset \mathbb{R}^d$ be a sequence of points. Define $K \triangleq \boldsymbol{k}_{\mathbb{P}}(\mathcal{S}_n, \mathcal{S}_n) \in \mathbb{R}^{n \times n}$. Suppose $\widehat{K} \in \mathbb{R}^{n \times n}$ is another PSD matrix such that $K \succeq \widehat{K}$. Define $D \triangleq \operatorname{diag}(K - \widehat{K})$, the diagonal part of $K - \widehat{K}$, and form $K' \triangleq \widehat{K} + D$. Let $w' \in \arg\min_{w \in \Delta_{n-1}} w'^{\top} K' w'$. Then for any $w \in \Delta_{n-1}$,*

$$\begin{aligned}\mathrm{MMD}^2_{\boldsymbol{k}_{\mathbb{P}}}(\mathbb{S}_n^w, \mathbb{P}) \leq \mathrm{MMD}^2_{\boldsymbol{k}_{\mathbb{P}}}(\mathbb{S}_n^{w_{\mathrm{OPT}}}, \mathbb{P}) + \operatorname{tr}((K - \widehat{K})^w) + \max_{i \in [n]}(K - \widehat{K})_{ii} \\ + (w^{\top} K' w - w'^{\top} K' w'). \end{aligned} \tag{3.46}$$

Proof of Lem. 3.F.2. Since $K = K' + (K - \widehat{K}) - D$ by construction, we have

$$\begin{aligned} w^{\top} K w &= w^{\top} K' w + w^{\top}(K - \widehat{K}) w - w^{\top} D w \\ &\leq w^{\top} K' w + w^{\top}(K - \widehat{K}) w \\ &= (w^{\top} K' w - w'^{\top} K' w') + w'^{\top} K' w' + w^{\top}(K - \widehat{K}) w \\ &\leq (w^{\top} K' w - w'^{\top} K' w') + w'^{\top} K' w' + \operatorname{tr}((K - \widehat{K})^w), \end{aligned}$$

where we used the fact that $D \succeq 0$ and Lem. 3.F.1. Next, by the definition of w', we have

$$\begin{aligned} w'^{\top} K' w' &\leq (w_{\mathrm{OPT}})^{\top} K' w_{\mathrm{OPT}} = (w_{\mathrm{OPT}})^{\top}(K' - K) w_{\mathrm{OPT}} + (w_{\mathrm{OPT}})^{\top} K w_{\mathrm{OPT}} \\ &= (w_{\mathrm{OPT}})^{\top}(D - (K - \widehat{K})) w_{\mathrm{OPT}} + (w_{\mathrm{OPT}})^{\top} K w_{\mathrm{OPT}} \\ &\leq (w_{\mathrm{OPT}})^{\top} D w_{\mathrm{OPT}} + (w_{\mathrm{OPT}})^{\top} K w_{\mathrm{OPT}} \\ &\leq \max_{i \in [n]}(K - \widehat{K})_{ii} + (w_{\mathrm{OPT}})^{\top} K w_{\mathrm{OPT}}, \end{aligned}$$

where we used the fact $K \succeq \widehat{K}$ in the penultimate step and Lem. 3.F.1 in the last step. Hence we have shown our claim. □

Lem. 3.F.2 shows that to control $\mathrm{MMD}^2_{\boldsymbol{k}_{\mathbb{P}}}(\mathbb{S}_n^w, \mathbb{P})$, it suffices to separately control the approximation error in terms of $\operatorname{tr}(K - \hat{K})$ and the optimization error $(w^{\top} K' w - w'^{\top} K' w')$. The next result establishes that using WeightedRPCholesky, we can obtain polynomial and exponential decay bounds for $\operatorname{tr}(K - \hat{K})$ in expectation depending on the kernel growth of $\boldsymbol{k}_{\mathbb{P}}$.

Proposition 3.F.1 (Approximation error of WeightedRPCholesky)**.** *Let $\boldsymbol{k}$ be a kernel satisfying Assum. ($\boldsymbol{\alpha}, \boldsymbol{\beta}$)-kernel. Let $\mathcal{S}_\infty$ be an infinite sequence of points in $\mathbb{R}^d$. For any $w \in \Delta_{n-1}$, let F be the low-rank approximation factor output by* WeightedRPCholesky$(\boldsymbol{k}, \mathcal{S}_n, w, r)$.

Define $K \triangleq \boldsymbol{k}(\mathcal{S}_n, \mathcal{S}_n)$. *If* $r \geq (\frac{\mathfrak{C}_d R_n^\beta + 1}{\sqrt{\log 2}} + \sqrt{\log 2})^2 - \frac{1}{\log 2}$, *then, with the expectation taken over the randomness in* WeightedRPCholesky,

$$\mathbb{E}\left[\operatorname{tr}\left((K - FF^\top)^w\right)\right] \leq H_{n,r}, \tag{3.47}$$

where $H_{n,r}$ *is defined as*

$$H_{n,r} \triangleq \begin{cases} 8\sum_{\ell=\mathfrak{U}(r)}^n (\frac{L_{\boldsymbol{k}}(R_n)}{\ell})^{\frac{2}{\alpha}} & \textsc{PolyGrowth}(\alpha, \beta), \\ 8\sum_{\ell=\mathfrak{U}(r)}^n \exp(1 - (\frac{\ell}{L_{\boldsymbol{k}}(R_n)})^{\frac{1}{\alpha}}) & \textsc{LogGrowth}(\alpha, \beta), \end{cases} \tag{3.48}$$

for $L_{\boldsymbol{k}}$ *defined in* (3.8) *and*

$$\mathfrak{U}(r) \triangleq \left\lfloor \sqrt{\frac{r + \frac{1}{\log 2}}{\log 2}} - \frac{1}{\log 2} \right\rfloor. \tag{3.49}$$

Moreover, $H_{n,r}$ *satisfies the bounds in Thm. 3.4.*

Proof of Prop. 3.F.1. Recall the notation $L_{\boldsymbol{k}}(R_n) = \frac{\mathfrak{C}_d R_n^\beta}{\log 2}$ from (3.8). Define $q \triangleq \mathfrak{U}(r)$ so that q is the biggest integer for which $r \geq 2q + q^2 \log 2$. The lower bound assumption of r is chosen such that $q > L_{\boldsymbol{k}}(R_n) > 0$. By Chen et al. [Che+22b, Theorem 3.1] with $\epsilon = 1$, we have

$$\mathbb{E}\left[\operatorname{tr}\left((K - FF^\top)^w\right)\right] \leq 2\sum_{\ell=q+1}^n \lambda_\ell(K^w). \tag{3.50}$$

Since $q > L_k(R_n)$, we can apply Cor. 3.B.1 to bound $\lambda_\ell(K^w)$ for $\ell \geq q+1$ and obtain (3.47) since $H_{n,r}$ (3.48) is constructed to match the bounds when applying Cor. 3.B.1 to (3.50). It remains to justify the bounds for $H_{n,r}$ in Thm. 3.4.

If $\boldsymbol{k}$ is $\textsc{PolyGrowth}(\alpha, \beta)$, by Assum. ($\boldsymbol{\alpha}, \boldsymbol{\beta}$)-kernel we have $\alpha < 2$. Hence

$$\begin{aligned} H_{n,r} = 8\sum_{\ell=q}^n \left(\frac{L_{\boldsymbol{k}}(R_n)}{\ell}\right)^{\frac{2}{\alpha}} &\leq 8 L_{\boldsymbol{k}}(R_n)^{\frac{2}{\alpha}} \int_{q-1}^\infty \ell^{-\frac{2}{\alpha}} \mathrm{d}\ell = 8 L_{\boldsymbol{k}}(R_n)^{\frac{2}{\alpha}} (q-1)^{1-\frac{2}{\alpha}} \\ &= O\left(\sqrt{r}(\tfrac{R_n^{2\beta}}{r})^{\frac{1}{\alpha}}\right), \end{aligned}$$

where we used the fact that $\int_{q-1}^\infty \ell^{-\frac{2}{\alpha}} \mathrm{d}\ell = (q-1)^{1-\frac{2}{\alpha}}$ for $\alpha < 2$, $L_{\boldsymbol{k}}(R_n) = O(R_n^\beta)$, and $q = \Theta(\sqrt{r})$.

If $\boldsymbol{k}$ is $\textsc{LogGrowth}(\alpha, \beta)$, then

$$H_{n,r} = 8\sum_{\ell=q}^n \exp\left(1 - \left(\frac{\ell}{L_{\boldsymbol{k}}(R_n)}\right)^{\frac{1}{\alpha}}\right) = 8e\sum_{\ell=q}^n c^{\ell^{1/\alpha}} \leq 8e\int_{\ell=q-1}^\infty c^{\ell^{1/\alpha}},$$

where $c \triangleq \exp(-L_{\boldsymbol{k}}(R_n)^{-1/\alpha}) \in (0,1)$. Defining $m \triangleq -\log c > 0$ and $q' = q - 1$, we have

$$\int_{x=q'}^\infty c^{x^{1/\alpha}} \mathrm{d}x = \int_{x=q'}^\infty \exp(-m x^{1/\alpha}) \mathrm{d}x = \alpha q' (m q'^{1/\alpha})^{-\alpha} \Gamma(\alpha, m q'^{1/\alpha}) = \alpha m^{-\alpha} \Gamma(\alpha, m q'^{1/\alpha}), \tag{3.51}$$

where $\Gamma(\alpha, x) \triangleq \int_x^\infty t^{\alpha-1}e^{-t}\mathrm{d}t$ is the incomplete gamma function. Since $\alpha > 0$, by Pinelis [Pin20, Thm. 1.1], we have

$$\Gamma(\alpha, mq'^{1/\alpha}) \leq \tfrac{(mq'^{1/\alpha}+b)^\alpha - (mq'^{1/\alpha})^\alpha}{\alpha b} e^{-mq'^{1/\alpha}},$$

where b is a known constant depending only on α. By the equivalence of norms on $\mathbb{R}^2$, there exists $C_\alpha > 0$ such that $(x+y)^\alpha \leq C_\alpha(x^\alpha + y^\alpha)$ for any $x, y > 0$. Hence

$$\Gamma(\alpha, mq'^{1/\alpha}) \leq \tfrac{(mq'^{1/\alpha}+b)^\alpha}{\alpha b} e^{-mq'^{1/\alpha}} \leq \tfrac{C_\alpha(m^\alpha q' + b^\alpha)}{\alpha b} e^{-mq'^{1/\alpha}}.$$

Hence from (3.51) we deduce

$$\textstyle\sum_{\ell=q'}^\infty c^{\ell^{1/\alpha}} \leq C_\alpha(q'b^{-1} + b^{\alpha-1}m^{-\alpha})e^{-mq'^{1/\alpha}}. \tag{3.52}$$

Since $m = -\log c = L_{\boldsymbol{k}}(R_n)^{-1/\alpha}$, we can bound the exponent by

$$-mq'^{1/\alpha} = -(L_k(R_n)^{-1}q')^{1/\alpha} = -(\tfrac{q'\log 2}{\mathfrak{C}_d R_n^\beta})^{1/\alpha} \leq -(\tfrac{0.83\sqrt{r}-2.39}{\mathfrak{C}_d R_n^\beta})^{1/\alpha},$$

where we used the fact that $q'\log 2 = (q-1)\log 2 \geq (\sqrt{\frac{r+\frac{1}{\log 2}}{\log 2}} - \frac{1}{\log 2} - 2)\log 2 \geq 0.83\sqrt{r} - 2.39$. On the other hand, since $q' = q - 1 \geq L(R_n) = m^{-\alpha}$, we can absorb the $b^{\alpha-1}m^{-1}$ term in (3.52) into q and finally obtain the bounds for $H_{n,r}$ in Thm. 3.4. □

The last piece of our analysis involves bounding the optimization error $(w^\top K' w - w'^\top K' w')$ in (3.46).

Lemma 3.F.3 (AMD guarantee for debiasing)**.** *Let $K \in \mathbb{R}^{n\times n}$ be an SPSD matrix. Let $f(w) \triangleq w^\top K w$. Then the final iterate x_T of Nesterov's 1-memory method [WAL23, Algorithm 14] after T steps with objective function $f(w)$, norm $\|\cdot\| = \|\cdot\|_1$, distance-generating function $\phi(x) = \sum_{i=1}^n x_i \log x_i$, and initial point $w_0 = (\frac{1}{n}, \ldots, \frac{1}{n}) \in \Delta_{n-1}$ satisfies*

$$f(w_T) - f(w_{\mathrm{OPT}}) \leq \tfrac{16\log n \max_{i\in[n]} K_{ii}}{T^2},$$

where $w_{\mathrm{OPT}} \in \arg\min_{x\in\mathbb{R}^n} f(x)$.

Proof of Lem. 3.F.3. We apply Wang, Abernethy, and Levy [WAL23, Theorem 14]. Hence it remains to determine the smoothness constant $L > 0$ such that, for all $x, y \in \Delta_{n-1}$,

$$\|\nabla f(x) - \nabla f(y)\|_\infty \leq L\|x - y\|_1,$$

and an upper bound for the Bregman divergence $D_{w_0}^\phi(w_{\mathrm{OPT}}) = \sum_{i=1}^n w_{\mathrm{OPT}i}\log\frac{w_{\mathrm{OPT}i}}{(w_0)_i} = \sum_{i=1}^n w_{\mathrm{OPT}i}\log n w_{\mathrm{OPT}i}$. To determine L, note $\nabla f(w) = 2Kw$, so we have, for any $x, y \in$

Δ_{n-1},

$$\begin{aligned}\|\nabla f(x) - \nabla f(y)\|_\infty &= 2\,\|K(x-y)\|_\infty = 2\max_{i\in[n]} |K_{i,:}(x-y)| \\ &\le 2\max_{i\in[n]} \|K_{i,:}\|_\infty \|x-y\|_1 = 2\left(\max_{i\in[n]} K_{ii}\right)\|x-y\|_1 \\ &= 2\left(\max_{i\in[n]} K_{ii}\right)\|x-y\|_1,\end{aligned}$$

where we used the fact that the largest entry in an SPSD matrix appears on its diagonal. Thus we can take the smoothness constant to be

$$L = 2\max_{i\in[n]} K_{ii}.$$

To bound $D^{\phi}_{w_0}(w_{\mathrm{OPT}})$, note that by Jensen's inequality,

$$D^{\phi}_{w_0}(w) = \textstyle\sum_{i=1}^n w_i \log n w_i \le \log(\sum_{i=1}^n n w_i^2) = \log n + \log \|w\|_2^2 \le \log n,$$

where we used the fact that $\|w\|_2^2 \le \|w\|_1 = 1$ for $w \in \Delta_{n-1}$. □

With these tools in hand, we turn to the proof of Thm. 3.4. For the runtime of LD, it follows from the fact that WeightedRPCholesky takes $O((d_{\boldsymbol{k}_{\mathbb{P}}} + r)nr)$ time and one step of AMD takes $O(nr)$ time.

The error analysis is different for the first adaptive iteration and the ensuing adaptive iterations. Roughly speaking, we will show that the first adaptive iteration brings the MMD gap to the desired level, while the ensuing iterations do not introduce an excessive amount of error.

Step 1. Bound $\Delta\,\mathrm{MMD}_{\boldsymbol{k}_{\mathbb{P}}}(w^{(1)})$

Let $K \triangleq \boldsymbol{k}_{\mathbb{P}}(\mathcal{S}_n, \mathcal{S}_n)$ and F denote the low-rank approximation factor generated by WeightedRPCholesky. Denote $\widehat{K} \triangleq FF^\top$. Then $K' = \widehat{K} + \mathrm{diag}(K - \widehat{K})$. First, note that since $w^{(0)} = (\frac{1}{n}, \dots, \frac{1}{n})$, Resample returns $\tilde{w} = w^{(0)}$ with probability one. By Lem. 3.F.2, we have, using $\sqrt{a+b} \le \sqrt{a} + \sqrt{b}$ for $a, b \ge 0$ repeatedly and Lem. 3.F.1 that $\mathrm{tr}((K - \widehat{K})^w) \le \lambda_1(K - \widehat{K})$ and $\max_{i\in[n]}(K - \widehat{K})_{ii} \le \lambda_1(K - \widehat{K})$,

$$\begin{aligned}\mathrm{MMD}_{\boldsymbol{k}_{\mathbb{P}}}(\mathbb{S}_n^{w^{(1)}}, \mathbb{P}) &\le \mathrm{MMD}_{\boldsymbol{k}_{\mathbb{P}}}(\mathbb{S}_n^{w_{\mathrm{OPT}}}, \mathbb{P}) + \sqrt{2\lambda_1(K - \widehat{K})} + \sqrt{{w^{(1)}}^\top K' w^{(1)} - w'^\top K' w'} \\ &\le \mathrm{MMD}_{\boldsymbol{k}_{\mathbb{P}}}(\mathbb{S}_n^{w_{\mathrm{OPT}}}, \mathbb{P}) + \sqrt{2\lambda_1(K - \widehat{K})} + \sqrt{\tfrac{16\log n \|\boldsymbol{k}_{\mathbb{P}}\|_n}{T^2}},\end{aligned}$$

where we applied Lem. 3.F.1 and Lem. 3.F.3 in the last inequality. Fix $\delta \in (0,1)$. By Markov's inequality, we have

$$\Pr(\sqrt{\lambda_1(K - \widehat{K})} > \sqrt{\tfrac{\mathbb{E}[\lambda_1(K-\widehat{K})]}{\delta}}) \le \delta.$$

This means that with probability at least $1-\delta$, we have

$$\mathrm{MMD}_{\boldsymbol{k}_{\mathbb{P}}}(\mathbb{S}_n^{w^{(1)}}, \mathbb{P}) \leq \mathrm{MMD}_{\boldsymbol{k}_{\mathbb{P}}}(\mathbb{S}_n^{w_{\mathrm{OPT}}}, \mathbb{P}) + \sqrt{\frac{2\mathbb{E}\left[\lambda_1(K-\widehat{K})\right]}{\delta}} + \sqrt{\frac{16\log n \|\boldsymbol{k}_{\mathbb{P}}\|_n}{T^2}}.$$

Note that the lower bound condition on r in Assum. ($\boldsymbol{\alpha}$,$\boldsymbol{\beta}$)-params implies the lower bound condition in Prop. 3.F.1. Hence, by Prop. 3.F.1 with $w = (\frac{1}{n}, \dots, \frac{1}{n})$ and using the identity $\lambda_1(K-\widehat{K}) \leq \mathrm{tr}(K-\widehat{K})$ while noting that a factor of n appears, we have

$$\mathrm{MMD}_{\boldsymbol{k}_{\mathbb{P}}}(\mathbb{S}_n^{w^{(1)}}, \mathbb{P}) \leq \mathrm{MMD}_{\boldsymbol{k}_{\mathbb{P}}}(\mathbb{S}_n^{w_{\mathrm{OPT}}}, \mathbb{P}) + \sqrt{\frac{2nH_{n,r}}{\delta}} + \sqrt{\frac{16\|\boldsymbol{k}_{\mathbb{P}}\|_n \log n}{T^2}}.$$

Step 2. Bound the error of the remaining iterations

Fix $\delta > 0$. The previous step shows that, with probability at least $1 - \frac{\delta}{2}$,

$$\mathrm{MMD}_{\boldsymbol{k}_{\mathbb{P}}}(\mathbb{S}_n^{w^{(1)}}, \mathbb{P}) \leq \mathrm{MMD}_{\boldsymbol{k}_{\mathbb{P}}}(\mathbb{S}_n^{w_{\mathrm{OPT}}}, \mathbb{P}) + \sqrt{\frac{4nH_{n,r}}{\delta}} + \sqrt{\frac{16\|\boldsymbol{k}_{\mathbb{P}}\|_n \log n}{T^2}}.$$

Fix $q > 1$, and let $\tilde{w}$ be the resampled weight defined in the q-th iteration in Alg. 3.3. Without loss of generality, we assume $\tilde{w}_i > 0$ for all $i > 0$, since if $w_i = 0$ then index i is irrelevant for the rest of the algorithm. Thus, thanks to Resample, we have $\tilde{w}_i \geq 1/n$ for all $i \in [n]$. Let a/b denote the entry-wise division between two vectors. As in the previous step of the proof, we let $K \triangleq \boldsymbol{k}_{\mathbb{P}}(\mathcal{S}_n, \mathcal{S}_n)$, F be the low-rank factor output by WeightedRPCholesky($\boldsymbol{k}_{\mathbb{P}}, \mathcal{S}_n, \tilde{w}, r$), and $\widehat{K} = FF^\top$. For any $w \in \Delta_{n-1}$, recall the notation $K^w \triangleq \mathrm{diag}(\sqrt{w})K\,\mathrm{diag}(\sqrt{w})$. Then we have

$$\begin{aligned}
w^\top K w &= (w/\sqrt{\tilde{w}})^\top \mathrm{diag}(K^{\tilde{w}})(w/\sqrt{\tilde{w}}) \\
&= (w/\sqrt{\tilde{w}})^\top (\mathrm{diag}(\sqrt{\tilde{w}})\widehat{K}\,\mathrm{diag}(\sqrt{\tilde{w}})) + \mathrm{diag}(\sqrt{\tilde{w}})(K-\widehat{K})\,\mathrm{diag}(\sqrt{\tilde{w}})))(w/\sqrt{\tilde{w}}) \\
&= w^\top \widehat{K} w + (w/\sqrt{\tilde{w}})^\top (\mathrm{diag}(\sqrt{\tilde{w}})(K-\widehat{K})\,\mathrm{diag}(\sqrt{\tilde{w}})))(w/\sqrt{\tilde{w}}) \\
&\leq w^\top \widehat{K} w + \max_{i\in[n]}(1/\tilde{w}_i)\,\mathrm{tr}(\mathrm{diag}(\sqrt{\tilde{w}})(K-\widehat{K})\,\mathrm{diag}(\sqrt{\tilde{w}})) \\
&\leq w^\top \widehat{K} w + n\,\mathrm{tr}((K-\widehat{K})^{\tilde{w}}). \qquad (3.53)
\end{aligned}$$

Note that

$$K' = \widehat{K} + \mathrm{diag}(K-\widehat{K}) = K + (\widehat{K}-K) + \mathrm{diag}(K-\widehat{K}).$$

Since $K' \succeq \widehat{K}$, we have

$$w^{(q)\top} \widehat{K} w^{(q)} \leq w^{(q)\top} K' w^{(q)} \leq \tilde{w}^\top K' \tilde{w}, \qquad (3.54)$$

where the last inequality follows from the if conditioning at the end of Alg. 3.3. In addition,

$$\begin{aligned}
\tilde{w}^\top K' \tilde{w} &= \tilde{w}^\top (K + (\widehat{K}-K) + \mathrm{diag}(K-\widehat{K}))\tilde{w} \\
&\leq \tilde{w}^\top K \tilde{w} + \tilde{w}^\top \mathrm{diag}(K-\widehat{K})\tilde{w} \\
&= \tilde{w}^\top K \tilde{w} + \sqrt{\tilde{w}}^\top \mathrm{diag}((K-\widehat{K})^{\tilde{w}})\sqrt{\tilde{w}} \\
&\leq w^\top K \tilde{w} + \mathrm{tr}((K-\widehat{K})^{\tilde{w}}),
\end{aligned}$$

where we used the fact that $K \succeq \widehat{K}$ and $\left\|\sqrt{\tilde{w}}\right\|_2 = 1$. Plugging the previous inequality into (3.54) and then into (3.53) with $w = w^{(q)}$, we get

$$w^{(q)\top} K w^{(q)} \leq \tilde{w}^\top K \tilde{w} + (n+1)\operatorname{tr}((K - \widehat{K})^{\tilde{w}}). \tag{3.55}$$

Taking square-root on both sides using $\sqrt{a+b} \leq \sqrt{a} + \sqrt{b}$ for $a, b \geq 0$ and the triangle inequality, we get

$$\begin{aligned}\mathrm{MMD}_{\boldsymbol{k}_\mathbb{P}}(\mathbb{S}_n^{w^{(q)}}, \mathbb{P}) &\leq \mathrm{MMD}_{\boldsymbol{k}_\mathbb{P}}(\mathbb{S}_n^{\tilde{w}}, \mathbb{P}) + \sqrt{(n+1)\operatorname{tr}((K - \widehat{K})^{\tilde{w}})} \\ &\leq \mathrm{MMD}_{\boldsymbol{k}_\mathbb{P}}(\mathbb{S}_n^{w^{(q-1)}}, \mathbb{P}) + \mathrm{MMD}_{\boldsymbol{k}_\mathbb{P}}(\mathbb{S}_n^{w^{(q-1)}}, \mathbb{S}_n^{\tilde{w}}) + \sqrt{(n+1)\operatorname{tr}((K - \widehat{K})^{\tilde{w}})}.\end{aligned}$$

By Markov's inequality, we have

$$\begin{aligned}\Pr(\mathrm{MMD}_{\boldsymbol{k}_\mathbb{P}}(\mathbb{S}_n^{w^{(q-1)}}, \mathbb{S}_n^{\tilde{w}}) > \sqrt{\tfrac{4Q\mathbb{E}\left[\mathrm{MMD}^2_{\boldsymbol{k}_\mathbb{P}}(\mathbb{S}_n^{w^{(q-1)}}, \mathbb{S}_n^{\tilde{w}})\right]}{\delta}}) &\leq \tfrac{\delta}{4Q} \\ \Pr(\sqrt{\operatorname{tr}((K - \widehat{K})^{\tilde{w}})} > \sqrt{\tfrac{4Q\mathbb{E}\left[\operatorname{tr}((K-\widehat{K})^{\tilde{w}})\right]}{\delta}}) &\leq \tfrac{\delta}{4Q}.\end{aligned}$$

By Prop. 3.E.1(c), we have

$$\mathbb{E}\left[\mathrm{MMD}^2_{\boldsymbol{k}_\mathbb{P}}(\mathbb{S}_n^{w^{(q-1)}}, \mathbb{S}_n^{\tilde{w}})\right] = \mathbb{E}\left[\mathbb{E}\left[\mathrm{MMD}^2_{\boldsymbol{k}_\mathbb{P}}(\mathbb{S}_n^{w^{(q-1)}}, \mathbb{S}_n^{\tilde{w}}) \middle| w^{(q-1)}\right]\right] \leq \tfrac{\|\boldsymbol{k}_\mathbb{P}\|_n}{n}.$$

Thus by the union bound, with probability at least $1 - \frac{\delta}{2Q}$, using Prop. 3.F.1 (recall low-rank approximation $\widehat{K}$ is obtained using $\tilde{w}$), we have

$$\mathrm{MMD}_{\boldsymbol{k}_\mathbb{P}}(\mathbb{S}_n^{w^{(q)}}, \mathbb{P}) \leq \mathrm{MMD}_{\boldsymbol{k}_\mathbb{P}}(\mathbb{S}_n^{w^{(q-1)}}, \mathbb{P}) + \sqrt{\tfrac{4Q\|\boldsymbol{k}_\mathbb{P}\|_n}{n\delta}} + \sqrt{\tfrac{4Q(n+1)H_{n,r}}{\delta}}. \tag{3.56}$$

Finally, applying union bound and summing up the bounds for $q = 1, \ldots, Q$, we get, with probability at least $1 - \delta$,

$$\Delta\,\mathrm{MMD}_{\boldsymbol{k}_\mathbb{P}}(w^{(q)}) \leq \sqrt{\tfrac{2nH_{n,r}}{\delta}} + \sqrt{\tfrac{16\|\boldsymbol{k}_\mathbb{P}\|_n \log n}{T^2}} + (Q-1)\left(\sqrt{\tfrac{4Q\|\boldsymbol{k}_\mathbb{P}\|_n}{n\delta}} + \sqrt{\tfrac{4Q(n+1)H_{n,r}}{\delta}}\right).$$

This matches the stated asymptotic bound in Thm. 3.4. □

♢ 3.F.2 Thinning with KT-Compress++

For compression with target distribution $\mathbb{P}$, we modify the original KT-Compress++ algorithm of [SDM22, Ex. 6]: in Halve and Thin of Compress++, we use KT-SPLIT with kernel $\boldsymbol{k}_\mathbb{P}$ without KT-SWAP (so our version of Compress++ outputs 2^g coresets, each of size $\sqrt{n}$), followed by KT-Swap to obtain a size $\sqrt{n}$ coreset. We call the resulting thinning algorithm KT-Compress++. We show in Lem. 3.F.4 and Cor. 3.F.1 that KT-Compress++ satisfies an MMD guarantee similar to that of quadratic-time kernel thinning.

Algorithm 3.F.3 KT-Compress++ (modified Shetty, Dwivedi, and Mackey [SDM22, Alg. 2] to minimize MMD to $\mathbb{P}$)

Input: kernel $\boldsymbol{k}_{\mathbb{P}}$ with zero-mean under $\mathbb{P}$, input points $\mathcal{S}_n = (x_i)_{i\in[n]}$, multiplicity n' with $n' \in 4^{\mathbb{N}}$, weight $w \in \Delta_{n-1} \cap (\frac{\mathbb{N}_0}{n'})^n$, thinning parameter $\mathfrak{g}$, failure probability δ

$\mathtt{S} \leftarrow$ index sequence where $k \in [n]$ appears $n'w_k$ times

$(\mathtt{I}^{(\ell)})_{\ell\in[2^{\mathfrak{g}}]} \leftarrow$ Compress++$(\mathfrak{g}, \mathcal{S}_n[\mathtt{S}])$ ▷ *Shetty, Dwivedi, and Mackey [SDM22, Ex. 6] with KT substituted with* KT-SPLIT *in* HALVE *and* THIN.

$\mathtt{I}^{(\ell)} \leftarrow \mathtt{S}[\mathtt{I}^{(\ell)}]$ for each $\ell \in [2^{\mathfrak{g}}]$

$\mathtt{I} \leftarrow$ KT-Swap$(\boldsymbol{k}_{\mathbb{P}}, \mathcal{S}_n, (\mathtt{I}^{(\ell)})_{\ell\in[2^{\mathfrak{g}}]})$

$w_{\mathrm{C++}} \leftarrow$ simplex weights corresponding to $\mathtt{I}$ ▷ $w_i = \frac{\text{number of occurrences of } i \text{ in } \mathtt{I}}{|\mathtt{I}|}$

Return: $w_{\mathrm{C++}} \in \Delta_{n-1} \cap (\frac{\mathbb{N}_0}{\sqrt{n}})^n$ ▷ *Hence* $\|w_{\mathrm{C++}}\|_0 \le \sqrt{n}$

Lemma 3.F.4 (Sub-gaussian guarantee for Compress++)**.** *Let $\mathcal{S}_n$ be a sequence of n points with $n \in 4^{\mathbb{N}}$. For any $\delta \in (0,1)$ and integer $\mathfrak{g} \ge \lceil \log_2 \log(n+1) + 3.1 \rceil$, consider the* Compress++ *algorithm [SDM22, Algorithm 2] with thinning parameter $\mathfrak{g}$, halving algorithm $\textsc{Halve}^{(k)} \triangleq$ symmetrized[6]$(\textsc{kt-split}(\boldsymbol{k}, \cdot, 1, \frac{n_k^2}{4n2^{\mathfrak{g}}(\mathfrak{g}+(\beta_n+1)2^{\mathfrak{g}})}\delta))$ for an input of $n_k \triangleq 2^{\mathfrak{g}+1+k}\sqrt{n}$ points and $\beta_n \triangleq \log_2\left(\frac{n}{n_0}\right)$, and with thinning algorithm $\textsc{Thin} \triangleq \textsc{kt-split}(\boldsymbol{k}, \cdot, \mathfrak{g}, \frac{\mathfrak{g}}{\mathfrak{g}+(\beta_n+1)2^{\mathfrak{g}}}\delta)$. Then this instantiation of* Compress++ *compresses $\mathcal{S}_n$ to $2^{\mathfrak{g}}$ coresets $(\mathcal{S}^{(i)}_{\mathrm{out}})_{i\in[2^{\mathfrak{g}}]}$ of $\sqrt{n}$ points each. Denote the signed measure $\phi^{(i)} \triangleq \frac{1}{n}\sum_{x\in\mathcal{S}_n}\delta_x - \frac{1}{\sqrt{n}}\sum_{x\in\mathcal{S}^{(i)}_{\mathrm{out}}}\delta_x$. Then for each $i \in [2^{\mathfrak{g}}]$, on an event $\mathcal{E}^{(i)}_{\mathrm{equi}}$ with $\Pr(\mathcal{E}^{(i)}_{\mathrm{equi}}) \ge 1 - \frac{\delta}{2}$, $\phi^{(i)} = \tilde{\phi}^{(i)}$ for a random signed measure $\tilde{\phi}^{(i)}$ such that, for any $\delta' \in (0,1)$,*

$$\Pr\left(\left\|\tilde{\phi}^{(i)}\boldsymbol{k}\right\|_{\mathcal{H}_{\boldsymbol{k}}} \ge a'_n\left(1 + \sqrt{\log(\tfrac{1}{\delta'})}\right)\right) \le \delta',$$

where

$$a'_n = \tfrac{4}{\sqrt{n}}\left(2 + \sqrt{\tfrac{8}{3}\|\boldsymbol{k}\|_n \log(\tfrac{6\sqrt{n}(\mathfrak{g}+(\frac{\log_2 n}{2}-\mathfrak{g})2^{\mathfrak{g}})}{\delta})\log(4\mathcal{N}_{\boldsymbol{k}}(\mathcal{B}_2(R_n), n^{-1/2}))}\right).$$

Proof of Lem. 3.F.4. This proof is similar to the one for Shetty, Dwivedi, and Mackey [SDM22, Ex. 6] but with explicit constant tracking and is self-contained, invoking only Shetty, Dwivedi, and Mackey [SDM22, Thm. 4] which gives MMD guarantees for Compress++ given the sub-Gaussian parameters of HALVE and THIN.

Recall that n_k is the number of input points for the halving subroutine at recursion level k in Compress++, and β_n is the total number of recursion levels. Let $\mathcal{S}_{\mathrm{C}}$ denote

[6] Any halving algorithm can be converted into an unbiased one by symmetrization, i.e., returning either the output half or its complement with equal probability [SDM22, Remark 3].

the output of COMPRESS [SDM22, Alg. 1] of size $2^{\mathfrak{g}}\sqrt{n}$. Fix $\delta, \delta' \in (0,1)$. Suppose we use $\text{HALVE}^{(k)} \triangleq \text{symmetrized}(\text{KT-SPLIT}(\boldsymbol{k}, \cdot, 1, \gamma_k\delta))$ for an input of n_k points for γ_k to be determined. Suppose we use $\text{THIN} \triangleq \text{KT-SPLIT}(\boldsymbol{k}, \cdot, \mathfrak{g}, \gamma'\delta)$ for γ' to be determined; this is the kernel thinning stage that thins $2^{\mathfrak{g}}\sqrt{n}$ points to $2^{\mathfrak{g}}$ coresets, each with $\sqrt{n}$ points. Since the analysis is the same for all coresets, we will fix an arbitrary coreset without superscript in the notation.

By Lem. 3.D.1, with notation $t \triangleq \log \frac{1}{\delta'}$, there exist events $\mathcal{E}_{k,j}$, $\mathcal{E}_{\mathrm{T}}$, and random signed measures $\phi_{k,j}$, $\tilde{\phi}_{k,j}$, ϕ_{T}, $\tilde{\phi}_{\mathrm{T}}$ for $0 \le k \le \beta_n$ and $j \in [4^k]$ such that

(a) $\Pr(\mathcal{E}_{k,j}^c) \le \frac{\gamma_k\delta}{2}$ and $\Pr(\mathcal{E}_{\mathrm{T}}^c) \le \frac{\gamma'\delta}{2}$,

(b) $\mathbf{1}_{\mathcal{E}_{k,j}}\phi_{k,j} = \mathbf{1}_{\mathcal{E}_{k,j}}\tilde{\phi}_{k,j}$ and $\mathbf{1}_{\mathcal{E}_{\mathrm{T}}}\phi_{\mathrm{T}} = \mathbf{1}_{\mathcal{E}_{\mathrm{T}}}\tilde{\phi}_{\mathrm{T}}$,

(c) We have

$$\Pr\left(\left\|\tilde{\phi}_{k,j}\boldsymbol{k}\right\|_{\mathcal{H}_{\boldsymbol{k}}} \ge a_{n_k} + v_{n_k}\sqrt{t}\middle|\{\tilde{\phi}_{k',j'}\}_{k'>k,j'\ge 1}, \{\tilde{\phi}_{k',j'}\}_{k',j'<j}\right) \le e^{-t}$$
$$\Pr\left(\left\|\tilde{\phi}_{\mathrm{T}}\boldsymbol{k}\right\|_{\mathcal{H}_{\boldsymbol{k}}} \ge a'_n + v'_n\sqrt{t}\middle|S_{\mathrm{C}}\right) \le e^{-t},$$

where, by Lem. 3.D.1, and by increasing the sub-Gaussian constants if necessary, we have

$$a_{n_k} \triangleq v_{n_k} \triangleq a_{n_k,n_k/2} = \frac{2}{n_k}\left(2 + \sqrt{\tfrac{8}{3}\|\boldsymbol{k}\|_n \log(\tfrac{3n_k}{\gamma_k\delta})\log(4\mathcal{N}_{\boldsymbol{k}}\left(\mathcal{B}_2(R_n), \tfrac{2}{n_k}\right))}\right),$$
$$a'_n \triangleq v'_n \triangleq a_{2^{\mathfrak{g}}\sqrt{n},\sqrt{n}} = \frac{1}{\sqrt{n}}\left(2 + \sqrt{\tfrac{8}{3}\|\boldsymbol{k}\|_n \log(\tfrac{6\mathfrak{g}\sqrt{n}}{\gamma'\delta})\log(4\mathcal{N}_{\boldsymbol{k}}\left(\mathcal{B}_2(R_n), n^{-1/2}\right))}\right), \quad \text{and}$$

(d) $\mathbb{E}\left[\tilde{\phi}_{k,j}\boldsymbol{k}\middle|\{\tilde{\phi}_{k',j'}\}_{k'>k,j'\ge 1}, \{\tilde{\phi}_{k',j'}\}_{k',j'<j}\right] = 0.$

Hence on the event $\mathcal{E} = \bigcap_{k,j}\mathcal{E}_{k,j} \cap \mathcal{E}_{\mathrm{T}}$, these properties hold simultaneously. We will choose $\{\gamma_k\}_k$ and γ' such that $\Pr(\mathcal{E}^c) \le \frac{\delta}{2}$. By the union bound,

$$\Pr(\mathcal{E}^c) \le \Pr(\mathcal{E}_{\mathrm{T}}^c) + \textstyle\sum_{k=0}^{\beta_n}\sum_{j=1}^{4^k}\Pr(\mathcal{E}_{k,j}^c) \le \frac{\gamma'\delta}{2} + \sum_{k=0}^{\beta_n} 4^k\frac{\gamma_k\delta}{2}. \tag{3.57}$$

On the event $\mathcal{E}$, we apply Shetty, Dwivedi, and Mackey [SDM22, Thm. 4, Rmk. 7] to get a sub-Gaussian guarantee for $\text{MMD}_{\boldsymbol{k}}(\mathbb{S}_n, \mathbb{S}_{\text{out}})$. We want to choose γ_k, γ' such that the rescaled quantities $\tilde{\zeta}_{\mathrm{H}} \triangleq \frac{n_0}{2}a_{n_0}$ and $\tilde{\zeta}_{\mathrm{T}} \triangleq \sqrt{n}a'_n$ satisfy $\tilde{\zeta}_{\mathrm{H}} = \tilde{\zeta}_{\mathrm{T}}$ [SDM22, Eq. (13)], which implies that we need

$$\frac{3n_0}{\gamma_0\delta} = \frac{6\mathfrak{g}\sqrt{n}}{\gamma'\delta} \iff \frac{\gamma_0}{\gamma'} = \frac{2^{\mathfrak{g}}}{\mathfrak{g}}. \tag{3.58}$$

Hence if we take $\gamma' = \frac{\mathfrak{g}}{\mathfrak{g}+(\beta_n+1)2^{\mathfrak{g}}}$ and $\gamma_k = \frac{n_k^2}{4n2^{\mathfrak{g}}(\mathfrak{g}+(\beta_n+1)2^{\mathfrak{g}})}$, then (3.58) holds and the upper bound in (3.57) becomes $\frac{\delta}{2}$. Note that $n_k a_{n_k}$ is non-decreasing in n_k, so by Shetty, Dwivedi,

and Mackey [SDM22, Theorem 4, Remark 7], Compress++$(\delta, \mathfrak{g})$ outputs a signed measure ϕ that, on the event $\mathcal{E}$ with $\Pr(\mathcal{E}^c) \leq \frac{\delta}{2}$, equals another signed measure $\tilde{\phi}$ that satisfies, for any $\delta' \in (0,1)$,

$$\Pr(\left\|\tilde{\phi}\boldsymbol{k}\right\|_{\mathcal{H}_{\boldsymbol{k}}} \geq \hat{a}_n + \hat{v}_n\sqrt{\log(\tfrac{1}{\delta'})}) \leq \delta',$$

where $\hat{a}_n, \hat{v}_n$ satisfy $\max(\hat{a}_n, \hat{v}_n) \leq 4a'_n$ whenever $\mathfrak{g} \geq \lceil \log_2 \log(n+1) + 3.1 \rceil$. □

Corollary 3.F.1 (MMD guarantee for Compress++)**.** *Let $\mathcal{S}_\infty$ be an infinite sequence of points in $\mathbb{R}^d$ and $\boldsymbol{k}$ a kernel. For any $\delta \in (0,1)$ and $n \in \mathbb{N}$ such that $n \in 4^{\mathbb{N}}$, consider the* Compress++ *with the same parameters as in Lem. 3.F.4 with $\mathfrak{g} \geq \lceil \log_2 \log(n+1) + 3.1 \rceil$. Then for any $i \in [\sqrt{n}]$, with probability at least $1-\delta$,*

$$\mathrm{MMD}_{\boldsymbol{k}}(\mathbb{S}_n, \mathbb{S}^{(i)}_{\mathrm{out}}) \leq \tfrac{4}{\sqrt{n}}\left(2 + \sqrt{\tfrac{8}{3}\|\boldsymbol{k}\|_n \log(\tfrac{6\sqrt{n}(\mathfrak{g}+(\frac{\log_2 n}{2}-\mathfrak{g})2^{\mathfrak{g}})}{\delta}) \log(4\mathcal{N}_{\boldsymbol{k}}(\mathcal{B}_2(R_n), n^{-1/2}))}\right)\left(1 + \sqrt{\log\tfrac{2}{\delta}}\right).$$

Proof. After applying Lem. 3.F.4 with $\delta' = \frac{\delta}{2}$ and following the same argument as in the proof of Cor. 3.D.1, we have, with probability at least $1-\delta$,

$$\mathrm{MMD}_{\boldsymbol{k}}(\mathbb{S}_n, \mathbb{S}^{(i)}_{\mathrm{out}}) \leq a'_n\left(1 + \sqrt{\log\tfrac{2}{\delta}}\right).$$

Plugging in the expression of a'_n from Lem. 3.F.4 gives the claimed bound. □

♢ 3.F.3 Proof of Thm. 3.5: MMD guarantee for LSKT

First of all, the claimed runtime follows from the runtime of LD (Thm. 3.4), the $O(d_{\boldsymbol{k}_{\mathbb{P}}} 4^{\mathfrak{g}} n \log n)$ runtime of Compress++, and the $O(d_{\boldsymbol{k}_{\mathbb{P}}} n^{1.5})$ runtime of KT-Swap.

Without loss of generality assume $n \in 4^{\mathbb{N}}$. Fix $\delta \in (0,1)$. Let $w^\diamond$ denote the output of LD, and w^{sr} denote the output of Resample, both regarded as random variables. By Thm. 3.4, we have, with probability at least $1 - \frac{\delta}{3}$,

$$\mathrm{MMD}_{\boldsymbol{k}_{\mathbb{P}}}(\mathbb{S}^{w^\diamond}_n, \mathbb{P}) = \mathrm{MMD}_{\boldsymbol{k}_{\mathbb{P}}}(\mathbb{S}^{w_{\mathrm{OPT}}}_n, \mathbb{P}) + O\left(\sqrt{\tfrac{nH_{n,r}}{\delta}}\right) + O\left(\sqrt{\tfrac{\|\boldsymbol{k}_{\mathbb{P}}\|_n \max(\log n, 1/\delta)}{n}}\right). \quad (3.59)$$

By Prop. 3.E.1(c) with $\boldsymbol{k} = \boldsymbol{k}_{\mathbb{P}}$, we have the upper bound

$$\mathbb{E}\left[\mathrm{MMD}^2_{\boldsymbol{k}_{\mathbb{P}}}(\mathbb{S}^{w^{\mathrm{sr}}}_n, \mathbb{S}^{w^\diamond}_n)\right] = \mathbb{E}\left[\mathbb{E}\left[\mathrm{MMD}^2_{\boldsymbol{k}_{\mathbb{P}}}(\mathbb{S}^{w^{\mathrm{sr}}}_n, \mathbb{S}^{w^\diamond}_n)\middle| w^\diamond\right]\right] \leq \tfrac{\|\boldsymbol{k}_{\mathbb{P}}\|_n}{n}.$$

Thus, by Markov's inequality,

$$\Pr(\mathrm{MMD}_{\boldsymbol{k}_{\mathbb{P}}}(\mathbb{S}^{w^{\mathrm{sr}}}_n, \mathbb{S}^{w^\diamond}_n) \geq \sqrt{\tfrac{3\|\boldsymbol{k}_{\mathbb{P}}\|_n}{n\delta}}) \leq \tfrac{\delta}{3}.$$

Hence, with probability at least $1 - \frac{\delta}{3}$, we have

$$\mathrm{MMD}_{\boldsymbol{k}_{\mathbb{P}}}(\mathbb{S}_n^{w^{\mathrm{sr}}}, \mathbb{S}_n^{w^{\diamond}}) \leq \sqrt{\frac{3\|\boldsymbol{k}_{\mathbb{P}}\|_n}{n\delta}}. \tag{3.60}$$

Let $\mathbb{S}_{\mathrm{out}}^{(i)}$ denote the i-th coreset output by Thin in KT-Compress++ (Alg. 3.F.3). By Cor. 3.F.1 with $\boldsymbol{k} = \boldsymbol{k}_{\mathbb{P}}$, we have, with probability at least $1 - \frac{\delta}{3}$,

$$\mathrm{MMD}_{\boldsymbol{k}_{\mathbb{P}}}(\mathbb{S}_n^{w^{\mathrm{sr}}}, \mathbb{S}_{\mathrm{out}}^{(i)}) = O\left(\sqrt{\frac{\|\boldsymbol{k}_{\mathbb{P}}\|_n \log n \log(e\mathcal{N}_{\boldsymbol{k}_{\mathbb{P}}}(\mathcal{B}_2(R_n), n^{-1/2}))}{n}} \log \tfrac{e}{\delta}\right).$$

Since KT-Swap can never increase $\mathrm{MMD}_{\boldsymbol{k}_{\mathbb{P}}}(\cdot, \mathbb{P})$, we have, by the triangle inequality,

$$\begin{aligned}\mathrm{MMD}_{\boldsymbol{k}_{\mathbb{P}}}(\mathbb{S}_n^{w_{\mathrm{LSKT}}}, \mathbb{P}) &\leq \mathrm{MMD}_{\boldsymbol{k}_{\mathbb{P}}}(\mathbb{S}_{\mathrm{out}}^{(1)}, \mathbb{P}) \\ &\leq \mathrm{MMD}_{\boldsymbol{k}_{\mathbb{P}}}(\mathbb{S}_{\mathrm{out}}^{(1)}, \mathbb{S}_n^{w^{\mathrm{sr}}}) + \mathrm{MMD}_{\boldsymbol{k}_{\mathbb{P}}}(\mathbb{S}_n^{w^{\mathrm{sr}}}, \mathbb{S}_n^{w^{\diamond}}) + \mathrm{MMD}_{\boldsymbol{k}_{\mathbb{P}}}(\mathbb{S}_n^{w^{\diamond}}, \mathbb{P}). \end{aligned} \tag{3.61}$$

By the union bound, with probability at least $1 - \delta$, the bounds (3.59), (3.60), (3.61) hold, so that the claim is shown by adding together the right-hand sides of these bounds and applying Assum. $(\boldsymbol{\alpha}, \boldsymbol{\beta})$-kernel. □

■ 3.G Simplex-Weighted Debiased Compression

In this section, we provide deferred analyses for RT and SR/LSR, as well as the algorithmic details of Recombination (Alg. 3.G.1) and KT-Swap-LS (Alg. 3.G.2).

◇ 3.G.1 MMD guarantee for RT

Proposition 3.G.1 (RT guarantee)**.** *Under Assums. 3.1 and $(\boldsymbol{\alpha}, \boldsymbol{\beta})$-kernel, given $w \in \Delta_{n-1}$ and that $m \geq (\frac{\mathfrak{C}_d R_n^{\beta}+1}{\sqrt{\log 2}} + \sqrt{\log 2})^2 - \frac{1}{\log 2} + 1$, RecombinationThinning (Alg. 3.5) outputs $w_{\mathrm{RT}} \in \Delta_{n-1}$ with $\|w_{\mathrm{RT}}\|_0 \leq m$ in $O((d_{\boldsymbol{k}_{\mathbb{P}}} + m)nm + m^3 \log n)$ time such that with probability at least $1 - \delta$,*

$$\mathrm{MMD}_{\boldsymbol{k}_{\mathbb{P}}}(\mathbb{S}_n^{w_{\mathrm{RT}}}, \mathbb{P}) \leq \mathrm{MMD}_{\boldsymbol{k}_{\mathbb{P}}}(\mathbb{S}_n^{w}, \mathbb{P}) + \sqrt{\frac{2\|\boldsymbol{k}_{\mathbb{P}}\|_n}{n\delta}} + \sqrt{\frac{2nH_{n,m-1}}{\delta}}, \tag{3.62}$$

where $H_{n,r}$ is defined in (3.48).

Proof of Prop. 3.G.1. The runtime follows from the $O((d_{\boldsymbol{k}_{\mathbb{P}}} + m)nm)$ runtime of WeightedRPCholesky, the $O(d_{\boldsymbol{k}_{\mathbb{P}}} nm)$ runtime of KT-Swap-LS, and the $O(m^3 \log n)$ runtime of Recombination [Tch16] which dominates the $O(m^3)$ weight optimization step.

Recall $w' \in \Delta_{n-1}$ from RT. The formation of F in Alg. 3.5 is identical to the formation of F (with $r = m - 1$) in Alg. 3.3 for $q > 1$. Thus by (3.53) with $w = w'$, $K = \boldsymbol{k}_{\mathbb{P}}(\mathcal{S}_n, \mathcal{S}_n)$,

$$w'^{\top} K w' \leq w'^{\top} F F^{\top} w' + n \operatorname{tr}((K - FF^{\top})^{\tilde{w}}),$$

where $K = \boldsymbol{k}_{\mathbb{P}}(\mathcal{S}_n, \mathcal{S}_n)$. By construction of w' using Recombination, we have $F^\top \tilde{w} = F^\top w'$. Since $K \succeq FF^\top$, we have

$$w'^\top K w' \leq \tilde{w}^\top FF^\top \tilde{w} + n \operatorname{tr}((K - FF^\top)^{\tilde{w}}) \leq \tilde{w}^\top K \tilde{w} + n \operatorname{tr}((K - FF^\top)^{\tilde{w}}).$$

We recognize the right-hand side is precisely the right-hand side of (3.55) aside from having a multiplier of n instead of $n+1$ in front of the trace and that F is rank $m-1$. Now applying (3.56) with $Q = \frac{1}{2}$, $w^{(q)} = w'$, $w^{(q-1)} = w$, $r = m-1$, and noticing that KT-Swap-LS and the quadratic-programming solve at the end cannot decrease the objective, we obtain (3.62) with probability at least $1-\delta$. Note that the lower bound of m in Assum. ($\boldsymbol{\alpha}$,$\boldsymbol{\beta}$)-params makes $r = m-1$ satisfy the lower bound for r in Prop. 3.F.1. □

♢ 3.G.2 Proof of Thm. 3.6: MMD guarantee for SR/ LSR

The claimed runtime follows from the runtime of SteinThinning (Alg. 3.D.1) or LD (Thm. 3.4) plus the runtime of RT (Prop. 3.G.1).

Note the lower bound for m in Assum. ($\boldsymbol{\alpha}$,$\boldsymbol{\beta}$)-params implies the lower bound condition in Prop. 3.G.1. For the case of SR, we proceed as in the proof of Thm. 3.3 and use Prop. 3.G.1. For the case of LSR, we proceed as in the proof of Thm. 3.5 and use Thm. 3.4 and Prop. 3.G.1. □

■ 3.H Constant-Preserving Debiased Compression

In this section, we provide deferred analyses for CT and SC/LSC.

♢ 3.H.1 MMD guarantee for CT

Proposition 3.H.1 (CT guarantee)**.** *Under Assums. 3.1 and ($\boldsymbol{\alpha}, \boldsymbol{\beta}$)-kernel, given $w \in \Delta_{n-1}$ and $m \geq (\frac{\mathfrak{C}_d R_n^\beta + 1}{\sqrt{\log 2}} + \frac{2}{\sqrt{\log 2}})^2 - \frac{1}{\log 2}$, CT outputs $w_{\mathrm{CT}} \in \mathbb{R}^n$ with $\mathbf{1}_n^\top w_{\mathrm{CT}} = 1$ and $\|w_{\mathrm{CT}}\|_0 \leq m$ in $O((d_{\boldsymbol{k}_{\mathbb{P}}} + m)nm + m^3)$ time such that, for any $\delta \in (0,1)$, with probability $1-\delta$,*

$$\mathrm{MMD}_{\boldsymbol{k}_{\mathbb{P}}}(\mathbb{S}_n^{w_{\mathrm{CT}}}, \mathbb{P}) \leq 2\,\mathrm{MMD}_{\boldsymbol{k}_{\mathbb{P}}}(\mathbb{S}_n^{w}, \mathbb{P}) + \sqrt{\tfrac{4H_{n,m'}}{\delta}},$$

where $H_{n,m}$ is defined in (3.48) and $m' \triangleq m + \log 2 - 2\sqrt{m \log 2 + 1}$.

Proof of Prop. 3.H.1. The runtime follows from the $O((d_{\boldsymbol{k}_{\mathbb{P}}} + m)nm)$ runtime of WeightedRPCholesky, the $O(nm)$ runtime of KT-Swap-LS, and the $O(m^3)$ runtime of matrix inversion in solving the two minimization problems using (3.66).

To improve the clarity of notation, we will use $w^\diamond$ to denote the input weight w to CT. For index sequences $\mathtt{I}, \mathtt{J} \subset [n]$ and a kernel $\boldsymbol{k}$, we use $\boldsymbol{k}(\mathtt{I}, \mathtt{J})$ to indicate the matrix

$\boldsymbol{k}(\mathcal{S}_n[\mathtt{I}], \mathcal{S}_n[\mathtt{J}]) = [\boldsymbol{k}(x_i, x_j)]_{i\in\mathtt{I}, j\in\mathtt{J}}$, and similarly for a function $f : \mathbb{R}^n \to \mathbb{R}$, we use $f(\mathtt{I})$ to denote the vector $(f(x_i))_{i\in\mathtt{I}}$.

Recall the regularized kernel is $\boldsymbol{k}_c \triangleq \boldsymbol{k}_{\mathbb{P}} + c$. Suppose for now that $c > 0$ is an arbitrary constant. Let $\mathtt{I}$ denote the indices output by WeightedRPCholesky in CT. Let

$$w^c \triangleq \arg\min_{w:\mathrm{supp}(w)\subset\mathtt{I}} \mathrm{MMD}^2_{\boldsymbol{k}_c}(\mathbb{S}^w_n, \mathbb{S}^{w^\diamond}_n).$$

Note that w^c is not a probability vector and may not sum to 1.

Step 1. Bound $\mathrm{MMD}^2_{\boldsymbol{k}_c}(\mathbb{S}^{w^c}_n, \mathbb{S}^{w^\diamond}_n)$ in terms of WeightedRPCholesky approximation error

We start by using an argument similar to that of Epperly and Moreno [EM24, Prop. 3] to exploit the optimality condition of w^c. Since

$$\arg\min_{w:\mathrm{supp}(w)\subset\mathtt{I}} \mathrm{MMD}^2_{\boldsymbol{k}_c}(\mathbb{S}^w_n, \mathbb{S}^{w^\diamond}_n) = \arg\min_{w:\mathrm{supp}(w)\subset\mathtt{I}} w_{\mathtt{I}}^\top \boldsymbol{k}_c(\mathtt{I},\mathtt{I}) w_{\mathtt{I}} - 2w^{\diamond\top}\boldsymbol{k}_c(\mathcal{S}_n, \mathtt{I})w_{\mathtt{I}},$$

by optimality, w^c satisfies,

$$\boldsymbol{k}_c(\mathtt{I},\mathtt{I})w^c_{\mathtt{I}} = \mathbb{S}^{w^\diamond}_n \boldsymbol{k}_c(\mathtt{I}).$$

We comment that the index sequence $\mathtt{I}$ returned by WeightedRPCholesky makes $\boldsymbol{k}_c(\mathtt{I},\mathtt{I})$ invertible with probability 1: by the Guttman rank additivity formula of Schur complement [Zha06, Eq. (6.0.4)], each iteration of WeightedRPCholesky chooses a pivot with a non-zero diagonal and thus increases the rank of the low-rank approximation matrix, which is spanned by the columns of pivots, by 1. Hence

$$\begin{aligned}\mathbb{S}^{w^c}_n \boldsymbol{k}_c(\cdot) &= \boldsymbol{k}_c(\cdot, \mathcal{S}_n)w^c = \boldsymbol{k}_c(\cdot,\mathtt{I})w^c_{\mathtt{I}} = \boldsymbol{k}_c(\cdot,\mathtt{I})\boldsymbol{k}_c(\mathtt{I},\mathtt{I})^{-1}\boldsymbol{k}_c(\mathtt{I},\mathtt{I})w^c_{\mathtt{I}}\\ &= \boldsymbol{k}_c(\cdot,\mathtt{I})\boldsymbol{k}_c(\mathtt{I},\mathtt{I})^{-1}\mathbb{S}^{w^\diamond}_n\boldsymbol{k}_c(\mathtt{I}) = \mathbb{S}^{w^\diamond}_n \boldsymbol{k}_{c\mathtt{I}}(\cdot),\end{aligned}$$

where $\boldsymbol{k}_{c\mathtt{I}}(x,y) \triangleq \boldsymbol{k}_c(x,\mathtt{I})\boldsymbol{k}_c(\mathtt{I},\mathtt{I})^{-1}\boldsymbol{k}_c(\mathtt{I},y)$. Then

$$\mathrm{MMD}^2_{\boldsymbol{k}_c}(\mathbb{S}^{w^c}_n, \mathbb{S}^{w^\diamond}_n) = \left\|\mathbb{S}^{w^\diamond}_n\boldsymbol{k}_c - \mathbb{S}^{w^c}_n\boldsymbol{k}_c\right\|^2_{\boldsymbol{k}_c} = \left\|\mathbb{S}^{w^\diamond}_n\boldsymbol{k}_c - \mathbb{S}^{w^\diamond}_n\boldsymbol{k}_{c\mathtt{I}}\right\|^2_{\boldsymbol{k}_c} = w^{\diamond\top}(\boldsymbol{k}_c - \boldsymbol{k}_{c\mathtt{I}})(\mathcal{S}_n,\mathcal{S}_n)w^\diamond.$$

Recall the index set $\mathtt{I}$ consists of the m pivots selected by WeightedRPCholesky on the input matrix

$$K^\diamond_c \triangleq \boldsymbol{k}_c(\mathcal{S}_n,\mathcal{S}_n)^{w^\diamond}.$$

Define

$$\widehat{K^\diamond_c} \triangleq \boldsymbol{k}_{c\mathtt{I}}(\mathcal{S}_n,\mathcal{S}_n)^{w^\diamond}.$$

Thus, by Lem. 3.F.1,

$$\mathrm{MMD}^2_{\boldsymbol{k}_c}(\mathbb{S}_n^{w^c}, \mathbb{S}_n^{w^\diamond}) = w^{\diamond\top}(\boldsymbol{k}_c - \boldsymbol{k}_{c\mathrm{I}})(\mathcal{S}_n, \mathcal{S}_n)w^\diamond = \sqrt{w^\diamond}^\top (K_c^\diamond - \widehat{K}_c^\diamond)\sqrt{w^\diamond} \\ \leq \lambda_1(K_c^\diamond - \widehat{K}_c^\diamond) \leq \mathrm{tr}(K_c^\diamond - \widehat{K}_c^\diamond).$$

Step 2. Bound $\mathrm{tr}(K_c^\diamond - \widehat{K}_c^\diamond)$ using the trace bound of the unregularized kernel

Let $[\![A]\!]_r$ denote the best rank-r approximation of an SPSD matrix $A \in \mathbb{R}^{n\times n}$ in the sense that

$$[\![A]\!]_r \triangleq \arg\min_{\substack{X\in\mathbb{R}^{n\times n} \\ X=X^\top \\ A\succeq X\succeq 0 \\ \mathrm{rank}(X)\leq r}} \mathrm{tr}(A - X). \tag{3.63}$$

By the Eckart-Young-Mirsky theorem applied to symmetric matrices [Dax+14, Theorem 19], the solution to (3.63) is given by r-truncated eigenvalue decomposition of A, so that

$$\mathrm{tr}(A - [\![A]\!]_r) = \textstyle\sum_{\ell=r+1}^n \lambda_\ell(A).$$

Let $q \triangleq \mathfrak{U}(m)$ where $\mathfrak{U}$ is defined in (3.49), so that by Chen et al. [Che+22b, Thm. 3.1] with $\epsilon = 1$, we have

$$\mathbb{E}\left[\mathrm{tr}(K_c^\diamond - \widehat{K}_c^\diamond)\right] \leq 2\,\mathrm{tr}(K_c^\diamond - [\![K_c^\diamond]\!]_q).$$

We know one specific rank-q approximation of $K_c^\diamond$:

$$\widetilde{K}_c^\diamond \triangleq [\![K^\diamond]\!]_{q-1} + \mathrm{diag}(\sqrt{w^\diamond})c\mathbf{1}_n\mathbf{1}_n^\top\,\mathrm{diag}(\sqrt{w^\diamond}),$$

which satisfies

$$K_c^\diamond - \widetilde{K}_c^\diamond = K^\diamond + \mathrm{diag}(\sqrt{w^\diamond})c\mathbf{1}_n\mathbf{1}_n^\top\,\mathrm{diag}(\sqrt{w^\diamond}) - \widetilde{K}_c^\diamond = K^\diamond - [\![K^\diamond]\!]_{q-1}.$$

Thus by the variational definition in (3.63), we have

$$\mathrm{tr}(K_c^\diamond - [\![K_c^\diamond]\!]_q) \leq \mathrm{tr}(K_c^\diamond - \widetilde{K}_c^\diamond) = \mathrm{tr}(K^\diamond - [\![K^\diamond]\!]_{q-1}) = \textstyle\sum_{\ell=q}^n \lambda_\ell(K^\diamond).$$

Note the last bound does not depend on c. The tail sum of eigenvalues in the last expression is the same (up to a constant multiplier) as the one in (3.50) except for an off-by-1 difference in the summation index. A simple calculation shows that for $m' \triangleq m + \log 2 - 2\sqrt{m\log 2 + 1}$, we have $\mathfrak{U}(m') = \mathfrak{U}(m) - 1$. Another simple calculation shows that $m \geq (\frac{\mathfrak{C}_d R_n^\beta + 1}{\sqrt{\log 2}} + \frac{2}{\sqrt{\log 2}})^2 - \frac{1}{\log 2}$ implies that m' satisfies the lower bound requirement of r in Prop. 3.F.1. Thus, arguing as in the proof that follows (3.50), we get

$$\mathbb{E}\left[\mathrm{tr}(K_c^\diamond - \widehat{K}_c^\diamond)\right] \leq H_{n,m'}.$$

Thus so far we have shown

$$\mathbb{E}[\mathrm{MMD}^2_{\boldsymbol{k}_c}(\mathbb{S}_n^{w^c}, \mathbb{S}_n^{w^\diamond})] \leq \mathbb{E}\left[\mathrm{tr}(K_c^\diamond - \widehat{K_c^\diamond})\right] \leq H_{n,m'}.$$

By Markov's inequality, with probability at least $1-\delta$, we have

$$\mathrm{MMD}_{\boldsymbol{k}_c}(\mathbb{S}_n^{w^c}, \mathbb{S}_n^{w^\diamond}) \leq \sqrt{\tfrac{H_{n,m'}}{\delta}}.$$

Recall that $\mathrm{MMD}_{\boldsymbol{k}}(\mu,\nu) = \|(\mu-\nu)\boldsymbol{k}\|_{\boldsymbol{k}}$ for signed measures μ,ν. By the triangle inequality, we have

$$\begin{aligned}\mathrm{MMD}_{\boldsymbol{k}_c}(\mathbb{S}_n^{w^c}, \mathbb{P}) &\leq \mathrm{MMD}_{\boldsymbol{k}_c}(\mathbb{S}_n^{w^c}, \mathbb{S}_n^{w^\diamond}) + \mathrm{MMD}_{\boldsymbol{k}_c}(\mathbb{S}_n^{w^\diamond}, \mathbb{P}) \\ &= \mathrm{MMD}_{\boldsymbol{k}_c}(\mathbb{S}_n^{w^c}, \mathbb{S}_n^{w^\diamond}) + \mathrm{MMD}_{\boldsymbol{k}_\mathbb{P}}(\mathbb{S}_n^{w^\diamond}, \mathbb{P}),\end{aligned}$$

where we used that fact that $\sum_{i\in[n]} w_i^\diamond = 1$ to get the identity $\mathrm{MMD}_{\boldsymbol{k}_c}(\mathbb{S}^{w^\diamond}, \mathbb{P}) = \mathrm{MMD}_{\boldsymbol{k}_\mathbb{P}}(\mathbb{S}^{w^\diamond}, \mathbb{P})$. Hence, with probability at least $1-\delta$,

$$\mathrm{MMD}_{\boldsymbol{k}_c}(\mathbb{S}_n^{w^c}, \mathbb{P}) \leq \mathrm{MMD}_{\boldsymbol{k}_\mathbb{P}}(\mathbb{S}_n^{w^\diamond}, \mathbb{P}) + \sqrt{\tfrac{H_{n,m'}}{\delta}}. \tag{3.64}$$

Step 3. Incorporating sum-to-one constraint

We now turn w^c into a constant-preserving weight while not inflating the MMD by much. Define

$$w^1 \triangleq \arg\min_{w:\mathrm{supp}(w)\subset \mathtt{I}, \sum_{i\in[n]} w_i = 1} \mathrm{MMD}^2_{\boldsymbol{k}_\mathbb{P}}(\mathbb{S}_n^w, \mathbb{P}). \tag{3.65}$$

Note w^1 is the weight right before KT-Swap-LS step in CT. Let $K_\mathtt{I} = \boldsymbol{k}_\mathbb{P}(\mathtt{I},\mathtt{I})$. Let $\mathbf{1}_\mathtt{I}$ denote the $|\mathtt{I}|$-dimensional all-one veector. The Karush-Kuhm-Tucker condition [Gho+21, Sec. 4.7] applied to (3.65) implies that, the solution w^1 is a stationary point of the Lagrangian function

$$L(w_\mathtt{I}, \lambda) \triangleq w_\mathtt{I}^\top K_\mathtt{I} w_\mathtt{I} + \lambda(\mathbf{1}_\mathtt{I}^\top w_\mathtt{I} - 1).$$

Then $\nabla_{w_\mathtt{I}} L(w_\mathtt{I}^1, \lambda) = 0$ implies $2K_\mathtt{I} w_\mathtt{I}^1 - \lambda \mathbf{1}_\mathtt{I} = 0$, so $w_\mathtt{I}^1 = \frac{\lambda K_\mathtt{I}^{-1}\mathbf{1}_\mathtt{I}}{2}$. The Lagrangian multiplier λ is determined by the constraint $\mathbf{1}_\mathtt{I}^\top w_\mathtt{I} = 1$, so we find

$$w_\mathtt{I}^1 = \frac{K_\mathtt{I}^{-1}\mathbf{1}_\mathtt{I}}{\mathbf{1}_\mathtt{I}^\top K_\mathtt{I}^{-1}\mathbf{1}_\mathtt{I}}. \tag{3.66}$$

Define

$$w^{c,\mathbb{P}} \triangleq \arg\min_{w:\mathrm{supp}(w)\subset \mathtt{I}} \mathrm{MMD}^2_{\boldsymbol{k}_c}(\mathbb{S}_n^w, \mathbb{P}).$$

Since $w^{c,\mathbb{P}}$ is optimized to minimize $\mathrm{MMD}_{\boldsymbol{k}_c}$ to $\mathbb{P}$ on the same support as w^c, we have

$$\mathrm{MMD}_{\boldsymbol{k}_c}(\mathbb{S}_n^{w^{c,\mathbb{P}}}, \mathbb{P}) \le \mathrm{MMD}_{\boldsymbol{k}_c}(\mathbb{S}_n^{w^c}, \mathbb{P}).$$

The optimality condition for $w^{c,\mathbb{P}}$ is

$$(K_{\mathtt{I}} + c\mathbf{1}_{\mathtt{I}}\mathbf{1}_{\mathtt{I}}^\top)w - c\mathbf{1}_{\mathtt{I}} = 0,$$

and hence by the Sherman–Morrison formula,

$$w_{\mathtt{I}}^{c,\mathbb{P}} = (K_{\mathtt{I}} + c\mathbf{1}_{\mathtt{I}}\mathbf{1}_{\mathtt{I}}^\top)^{-1}c\mathbf{1}_{\mathtt{I}} = \left(K_{\mathtt{I}}^{-1} - \frac{cK_{\mathtt{I}}^{-1}\mathbf{1}_{\mathtt{I}}\mathbf{1}_{\mathtt{I}}^\top K_{\mathtt{I}}^{-1}}{1+c\mathbf{1}_{\mathtt{I}}^\top K_{\mathtt{I}}^{-1}\mathbf{1}_{\mathtt{I}}}\right)c\mathbf{1}_{\mathtt{I}} = \frac{K_{\mathtt{I}}^{-1}\mathbf{1}_{\mathtt{I}}}{1/c+\mathbf{1}_{\mathtt{I}}^\top K_{\mathtt{I}}^{-1}\mathbf{1}_{\mathtt{I}}}.$$

Let $\rho_c \triangleq \frac{\mathbf{1}_{\mathtt{I}}^\top K_{\mathtt{I}}^{-1}\mathbf{1}_{\mathtt{I}}}{1/c+\mathbf{1}_{\mathtt{I}}^\top K_{\mathtt{I}}^{-1}\mathbf{1}_{\mathtt{I}}}$, so that $w_{\mathtt{I}}^{c,\mathbb{P}} = \rho_c w_{\mathtt{I}}^1$. In particular, w^1 and $w^{c,\mathbb{P}}$ are scalar multiples of one another. To relate $\mathrm{MMD}_{\boldsymbol{k}_{\mathbb{P}}}(\mathbb{S}_n^{w^1}, \mathbb{P})$ and $\mathrm{MMD}_{\boldsymbol{k}_c}(\mathbb{S}_n^{w^{c,\mathbb{P}}}, \mathbb{P})$, note that

$$\begin{aligned}\mathrm{MMD}^2_{\boldsymbol{k}_{\mathbb{P}}}(\mathbb{S}_n^{w^1}, \mathbb{P}) = {w_{\mathtt{I}}^1}^\top K_{\mathtt{I}} w_{\mathtt{I}}^1 &= \frac{{w_{\mathtt{I}}^{c,\mathbb{P}}}^\top K_{\mathtt{I}} w_{\mathtt{I}}^{c,\mathbb{P}}}{\rho_c^2} = \frac{{w_{\mathtt{I}}^{c,\mathbb{P}}}^\top (K_{\mathtt{I}}+c\mathbf{1}_{\mathtt{I}}\mathbf{1}_{\mathtt{I}}^\top) w_{\mathtt{I}}^{c,\mathbb{P}} - c(\mathbf{1}_{\mathtt{I}}^\top w_{\mathtt{I}}^c)^2}{\rho_c^2} \\ &= \frac{\mathrm{MMD}^2_{\boldsymbol{k}_c}(\mathbb{S}_n^{w^{c,\mathbb{P}}}, \mathbb{P}) + 2c\mathbf{1}_{\mathtt{I}}^\top w_{\mathtt{I}}^c - c - c(\mathbf{1}_{\mathtt{I}}^\top w_{\mathtt{I}}^c)^2}{\rho_c^2} = \frac{\mathrm{MMD}^2_{\boldsymbol{k}_c}(\mathbb{S}_n^{w^{c,\mathbb{P}}}, \mathbb{P}) - c(\rho_c - 1)^2}{\rho_c^2}.\end{aligned}$$

So far the argument does not depend on any particular choice of $c > 0$. Let us now discuss how to choose c. Note that

$$\mathbf{1}_{\mathtt{I}}^\top K_{\mathtt{I}}^{-1}\mathbf{1}_{\mathtt{I}} = m\tfrac{\mathbf{1}_{\mathtt{I}}}{\sqrt{m}}^\top K_{\mathtt{I}}^{-1}\tfrac{\mathbf{1}_{\mathtt{I}}}{\sqrt{m}} \ge m\lambda_m(K_{\mathtt{I}}^{-1}) \ge \frac{m}{\lambda_1(K_{\mathtt{I}})} \ge \frac{m}{\mathrm{tr}(K_{\mathtt{I}})} \ge \frac{m}{\sum_{i\in[m]}\mathrm{diag}(K)_i^\downarrow},$$

where $\mathrm{diag}(K)^\downarrow$ denote the diagonal entries of $K = \boldsymbol{k}_{\mathbb{P}}(\mathcal{S}_n, \mathcal{S}_n)$ sorted in descending order. Thus

$$\rho_c = \frac{1}{\frac{1}{c\mathbf{1}_{\mathtt{I}}^\top K_{\mathtt{I}}^{-1}\mathbf{1}_{\mathtt{I}}}+1} \ge \frac{1}{\frac{\sum_{i\in[m]}\mathrm{diag}(K)_i^\downarrow}{mc}+1}.$$

Hence we can choose c to make sure ρ_c is bounded from below by a positive value. Recall in CT, we take

$$c = \frac{\sum_{i\in[m]}\mathrm{diag}(K)_i^\downarrow}{m},$$

so that $\rho_c \ge \frac{1}{2}$ and

$$\mathrm{MMD}^2_{\boldsymbol{k}_{\mathbb{P}}}(\mathbb{S}_n^{w^1}, \mathbb{P}) = \frac{\mathrm{MMD}^2_{\boldsymbol{k}_c}(\mathbb{S}_n^{w^{c,\mathbb{P}}}, \mathbb{P}) - c(\rho_c-1)^2}{\rho_c^2} \le 4\,\mathrm{MMD}^2_{\boldsymbol{k}_c}(\mathbb{S}_n^{w^{c,\mathbb{P}}}, \mathbb{P}).$$

Hence by (3.64) and the fact that KT-Swap-LS and the final reweighting in CT only improves MMD, we have, with probability at least $1-\delta$,

$$\begin{aligned}\mathrm{MMD}_{\boldsymbol{k}_{\mathbb{P}}}(\mathbb{S}_n^{w_{\mathrm{CT}}}, \mathbb{P}) \le \mathrm{MMD}_{\boldsymbol{k}_{\mathbb{P}}}(\mathbb{S}_n^{w^1}, \mathbb{P}) \le 2\,\mathrm{MMD}_{\boldsymbol{k}_c}(\mathbb{S}_n^{w^{c,\mathbb{P}}}, \mathbb{P}) &\le 2\,\mathrm{MMD}_{\boldsymbol{k}_c}(\mathbb{S}_n^{w^c}, \mathbb{P}) \\ &\le 2\,\mathrm{MMD}_{\boldsymbol{k}_{\mathbb{P}}}(\mathbb{S}_n^{w^\diamond}, \mathbb{P}) + 2\sqrt{\tfrac{H_{n,m'}}{\delta}},\end{aligned}$$

where we use (3.64) in the last inequality. □

♢ 3.H.2 Proof of Thm. 3.7: MMD guarantee for SC / LSC

The claimed runtime follows from the runtime of SteinThinning (Alg. 3.D.1) or LD (Thm. 3.4) plus the runtime of CT (Prop. 3.H.1).

Note the lower bound for m in Assum. ($\boldsymbol{\alpha}$,$\boldsymbol{\beta}$)-params implies the lower bound condition in Prop. 3.H.1. For the case of SC, we proceed as in the proof of Thm. 3.3 and use Prop. 3.H.1. For the case of LSC, we proceed as in the proof of Thm. 3.5 by invoking Thm. 3.4 and Prop. 3.H.1. □

■ 3.I Implementation and Experimental Details

♢ 3.I.1 $O(d)$-time Stein kernel evaluation

In this section, we show that for $\mathcal{S}_n = (x_i)_{i\in[n]}$, each Stein kernel evaluation $\boldsymbol{k}_p(x_i, x_j)$ for a radially analytic base kernel (Def. 3.B.3) can be done in $O(d)$ time after computing certain sufficient statistics in $O(nd^2 + d^3)$ time. Let $M \in \mathbb{R}^{d\times d}$ be a positive definite preconditioning matrix for $\boldsymbol{k}_p$. Let L be the Cholesky decomposition of M which can be done in $O(d^3)$ time so that $M = LL^\top$. From the expression (3.17), we can achieve $O(d)$ time evaluation if we can compute $\|x-y\|_M^2$ and $M\nabla \log p(x)$ in $O(d)$ time. For $M\nabla \log p(x)$, we can simply precompute $M\nabla \log p(x_i)$ for all $i \in [n]$. For $\|x-y\|_M^2$, we have

$$\|x-y\|_M^2 = (x-y)^\top M^{-1}(x-y) = (x-y)^\top (LL^\top)^{-1}(x-y) = \left\|L^{-1}(x-y)\right\|_2^2.$$

Hence it suffices to precompute $L^{-1}x_i$ for all $i \in [n]$, and we can precompute the inverse L^{-1} in $O(d^3)$ time.

♢ 3.I.2 Default parameters for algorithms

For LD, we always use $Q = 3$. To ensure that the guarantees of Lem. 3.F.3 and Thm. 3.4 hold while achieving fast convergence in practice, we take the step size of AMD to be $1/(8\|\boldsymbol{k}_{\mathbb{P}}\|_n)$ in the first adaptive round and $1/(8\sum_{i\in[n]} w_i^{(q-1)}\boldsymbol{k}_{\mathbb{P}}(x_i,x_i))$ in subsequent adaptive rounds. We use $T = 7\sqrt{n_0}$ for AMD in all experiments.

We implemented our modified versions of KernelThinning and KT-Compress++ in JAX [Bra+18] so that certain subroutines can achieve a speedup using just-in-time compilation and the parallel computation power of GPUs. For Compress++, we use $\mathfrak{g} = 4$ in all experiments as in Shetty, Dwivedi, and Mackey [SDM22]. For both KernelThinning and KT-Compress++, we use choose $\delta = 1/2$ as in the `goodpoints` library.

Each experiment was run with a single NVIDIA RTX 6000 GPU and an AMD EPYC 7513 32-Core CPU.

♢ 3.I.3 Correcting for burn-in details

We use the four MCMC chains provided by Riabiz et al. [Ria+22] that include both the sample points and their scores. The reference chain used to compute the energy distance is the same one used in Riabiz et al. [Ria+22] for the energy distance and was kindly provided by the authors.

In Tab. 3.I.1, we collect the runtime for the burn-in correction experiments.

Fig. 3.I.1, Fig. 3.I.2, Fig. 3.I.3, display the results of the burn-in correction experiment of Sec. 3.5 repeated with three other MCMC algorithms: MALA without preconditioning, random walk (RW), and adaptive random walk (ADA-RW). The results of P-MALA from Sec. 3.5 are also included for completeness. For all four chains, our methods reliably achieve better quality coresets when compared with the baseline methods.

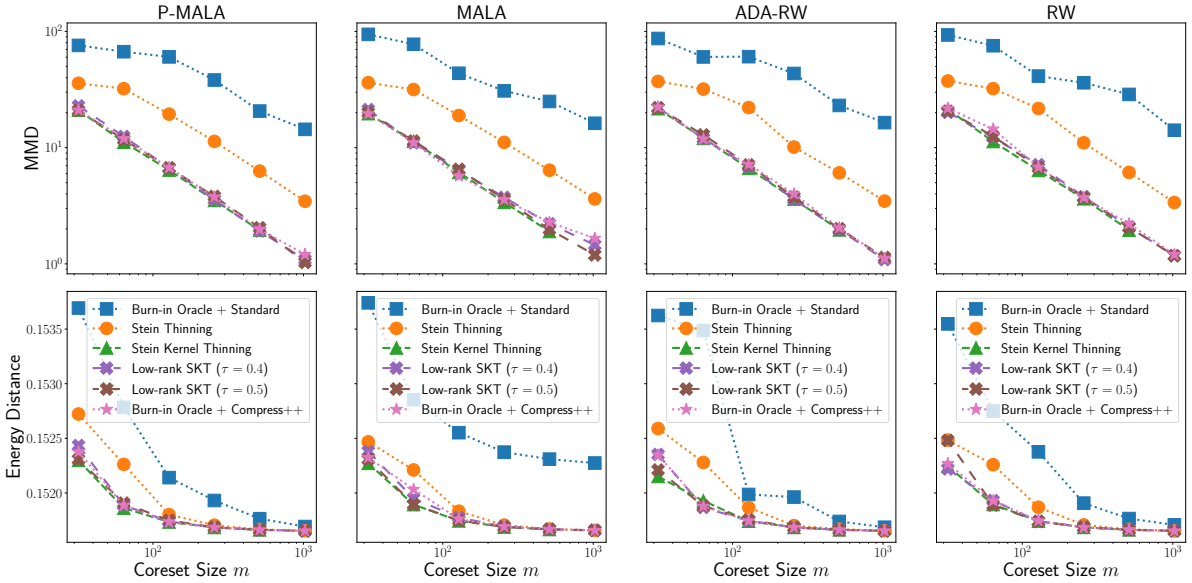

Figure 3.I.1: **Correcting for burn-in with equal-weighted compression.** For each of four MCMC algorithms and using only one chain, our methods consistently outperform the Stein and standard thinning baselines and match the 6-chain oracle.

♢ 3.I.4 Correcting for approximate MCMC details

Surrogate ground truth Following Liu and Lee [LL17], we took the first 10,000 data points and generated 2^{20} surrogate ground truth sample points using NUTS [HG+14] for the evaluation. To generate the surrogate ground truth using NUTS, we used `numpyro` [PPJ19]. It took 12 hours to generate the surrogate ground truth points using the GPU

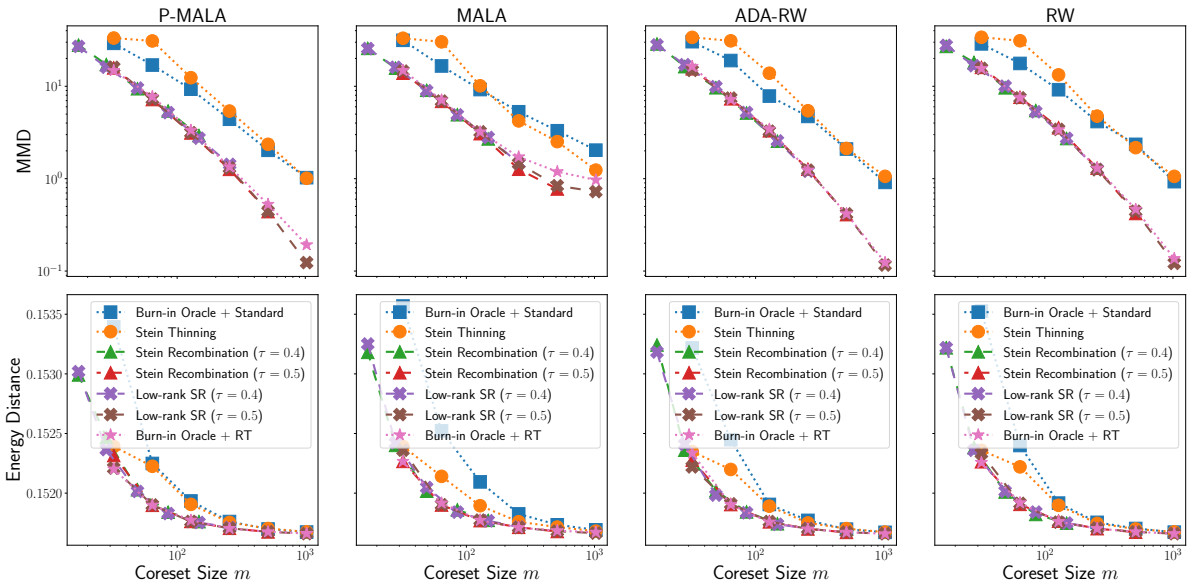

Figure 3.I.2: **Correcting for burn-in with simplex-weighted compression.** For each of four MCMC algorithms and using only one chain, our methods consistently outperform the Stein and standard thinning baselines and match the 6-chain oracle.

implementation, and we estimate it would have taken 200 hours using the CPU implementation.

SGFS For SGFS, we used batch size 32 and the step size schedule $\eta/(1+t)^{0.55}$ where t is the step count and η is the initial step size. We chose η from $\{10.0, 5.0, 1.0, 0.5, 0.1, 0.05, 0.01\}$, found $\eta = 1.0$ gave the best standard thinning MMD to get a coreset size of $m = 2^{10}$, and hence we fixed $\eta = 1.0$ in all experiments. We used the version of SGFS [AKW12, SGFS-f] that involves inversion of $d \times d$ matrices — we found the faster version (SGFS-d) that inverts only the diagonal resulted in significantly worse mixing. We implemented SGFS in `numpy` and ran it on the CPU.

Runtime The SGFS chain of length 2^{24} took approximately 2 hours to generate using the CPU. Remarkably, all of our low-rank methods finish within 10 minutes for $n_0 = 2^{20}$, which is orders of magnitude faster than the time taken to generate the NUTS surrogate ground truth.

Additional results In Fig. 3.I.4, we plot the posterior mean mean-squared error (MSE) for each compression method in the approximate MCMC experiment of Sec. 3.5.

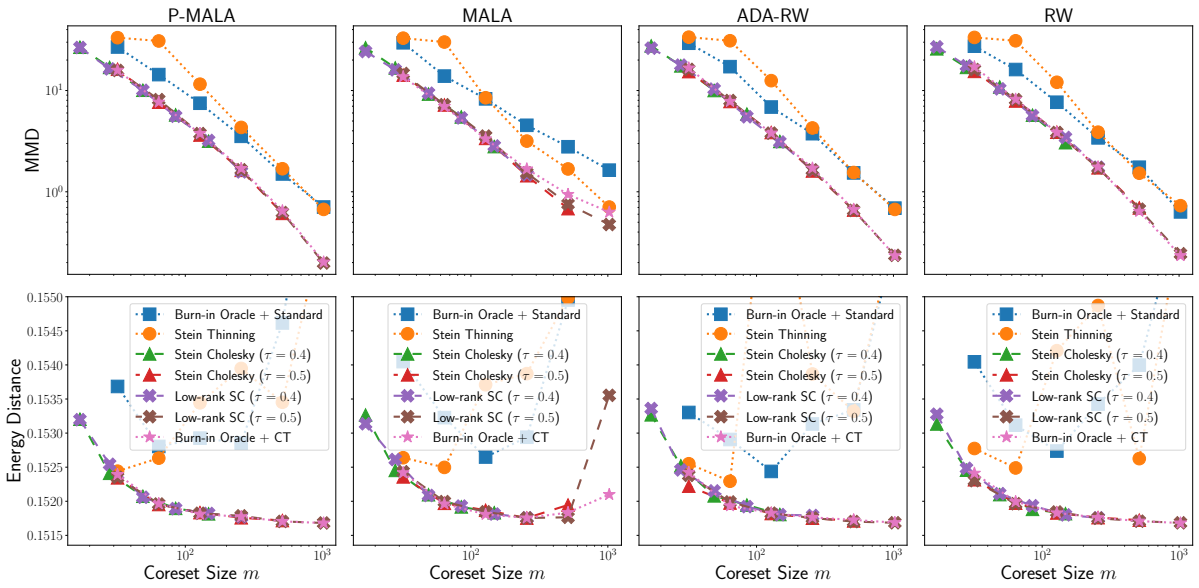

Figure 3.I.3: **Correcting for burn-in with constant-preserving compression.** For each of four MCMC algorithms and using only one chain, our methods consistently outperform the Stein and standard thinning baselines and match the 6-chain oracle.

♢ 3.I.5 Correcting for tempering details

In the data release of Riabiz et al. [Ria+20], we noticed there were 349 sample points for which the provided scores were NaNs, so we removed those points at the recommendation of the authors.

n_0	ST	LD (0.5)	LD (0.4)	KT	KT-Compress++	RT (0.5)	RT (0.4)	CT (0.5)	CT (0.4)
2^{14}	2.50	13.22	12.88	7.31	3.49	0.79	0.60	2.06	1.96
2^{16}	8.48	16.15	15.82	20.77	5.90	2.59	1.68	3.66	3.04
2^{18}	111.06	32.14	20.60	193.03	11.73	11.16	2.63	6.48	3.67
2^{20}	-	314.67	131.31	-	35.99	113.71	11.06	51.14	8.42

Table 3.I.1: **Breakdown of runtime (in seconds)** for the burn-in correction experiment ($d = 4$) of Sec. 3.5. n_0 is the input size after standard thinning from the length $n = 2 \times 10^6$ chain (Rem. 3.2). Each runtime is the median of 3 runs. KT and KT-Compress++ output $m = \sqrt{n_0}$ equal-weighted points. RT and CT respectively output $m = n_0^\tau$ points with simplex or constant-preserving weights for τ shown in parentheses. In addition, LD, RT, and CT use the rank n_0^τ. ST and KT took longer than 30 minutes for $n_0 = 2^{20}$ and hence their numbers are not reported.

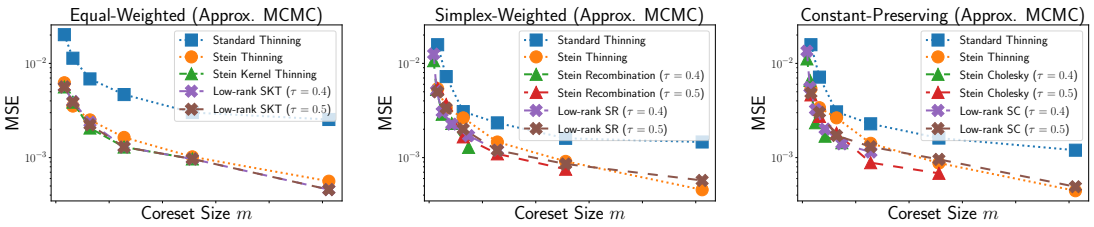

Figure 3.I.4: **Posterior mean mean-squared error (MSE)** for the approximate MCMC compression experiment of Sec. 3.5. MSE is computed as $\|\hat{\mathbb{E}}_{\mathbb{P}} Z - \sum_{i \in [n_0]} w_i x_i\|_M^2 / d$ where $\hat{\mathbb{E}}_{\mathbb{P}} Z$ is the mean of the surrogate ground truth NUTS sample.

Algorithm 3.G.1 Recombination (rephrasing of Tchernychova [Tch16, Alg. 1] that takes $O(m^3 \log n)$ time)

Input: matrix $A \in \mathbb{R}^{m\times n}$ with $m < n$ and one row of A all positive, a nonnegative vector $x_0 \in \mathbb{R}^n_{\geq 0}$.
function FindBFS(A, x_0)
▷ *The requirement of A and x_0 are the same as the input. This subroutine takes $O(n^3)$ time.*
$x \leftarrow x_0$
$U, S, V^\top \leftarrow \mathrm{SVD}(A)$ ▷ *any $O(n^3)$-time SVD algorithm that gives $USV^\top = A$*
$V \leftarrow (V^\top)_{m+1:n}$ ▷ *$V \in \mathbb{R}^{(n-m)\times n}$ so that the null space of A is spanned by the rows of V*
for $i = 1$ **to** $n - m$ **do**
$v \leftarrow V_i$
$k \leftarrow \arg\min_{j\in[n]:v_j>0} \frac{x_j}{v_j}$ ▷ *This must succeed because $Av = 0$ and A has an all-positive row, so one of the coordinates of v must be positive.*
$x \leftarrow x - \frac{x_k}{v_k} v$ ▷ *This zeros out the k-th coordinate of x while still ensuring x is nonnegative.*
for $j = i + 1$ **to** $n - m$ **do**
$V_j \leftarrow V_j - \frac{V_{j,k}}{v_k} v$ ▷ *$\{V_j\}_{j=i+1}^{n-m}$ remain independent and have 0 on the k-th coordinate.*
end for
end for
return: $x \in \mathbb{R}^n_{\geq 0}$ such that $Ax = Ax_0$ and $\|x\|_0 \leq m$.
end function
$x \leftarrow x_0$
while $\|x\|_0 > 2m$ **do**
Divide $\{i \in [n] : x_i > 0\}$ into $2m$ index blocks $\mathtt{I}_1, \ldots, \mathtt{I}_{2m}$, each of size at most $\left\lfloor \frac{\|x\|_0}{2m} \right\rfloor$.
$A_i \leftarrow A_{:,\mathtt{I}_i} x_{\mathtt{I}_i} \in \mathbb{R}^m, \forall i \in [2m]$
Form $\hat{A}$ to be the $m \times 2m$ matrix with columns A_i ▷ *Hence, one row of A contains all positive entries.*
$\hat{x} \leftarrow \mathrm{FindFBS}(\hat{A}, \mathbf{1}_{2m})$ ▷ *$\|\hat{x}\|_0 \leq n$ and $\hat{A}\hat{x} = \sum A_i \hat{x}_i = \sum A_i = \sum A_{:,\mathtt{I}_i} x_{\mathtt{I}_i} = Ax$.*
for $i = 1$ **to** $2m$ **do**
$x_{\mathtt{I}_i} \leftarrow \hat{x}_i \cdot x_{\mathtt{I}_i}$ **if** $\hat{x}_i > 0$ **else** 0
end for
▷ *After the update, the support of x shrinks by 2 while it maintains that $Ax = Ax_0$.*
end while
if $\|x\|_0 \geq m + 1$ **then**
$\mathtt{I} \leftarrow \{i \in [n] : x_i > 0\}$
$x_{\mathtt{I}} = \mathrm{FindBFS}(A_{:,\mathtt{I}}, x_{\mathtt{I}})$
end if
Return: $x \in \mathbb{R}^n_{\geq 0}$ such that $Ax = Ax_0$ and $\|x\|_0 \leq m$.

Algorithm 3.G.2 KT-Swap with Linear Search (KT-Swap-LS)

Input: kernel $\boldsymbol{k}_{\mathbb{P}}$ with zero-mean under $\mathbb{P}$, input points $\mathcal{S}_n = (x_i)_{i\in[n]}$, weights $w \in \Delta_{n-1}$, $\texttt{fmt} \in \{\text{SPLX}, \text{CP}\}$
$\texttt{S} \leftarrow \{i \in [n] : w_i \neq 0\}$
▷ *Maintain two sufficient statistics: $g = Kw$ and $D = w^\top Kw$.*
function $\text{Add}(g, D, i, t)$
  $g \leftarrow g + t\boldsymbol{k}_{\mathbb{P}}(\mathcal{S}_n, x_i)$
  $D \leftarrow D + 2tg_i + t^2\boldsymbol{k}_{\mathbb{P}}(x_i, x_i)$
  return: (g, D)
end function
function $\text{Scale}(g, D, \alpha)$
  $g \leftarrow \alpha g$
  $D \leftarrow \alpha^2 D$
  return: (g, D)
end function
$\texttt{Kdiag} \leftarrow \boldsymbol{k}_{\mathbb{P}}(\mathcal{S}_n, \mathcal{S}_n)$
$g \leftarrow \mathbf{0} \in \mathbb{R}^n$
$D \leftarrow 0$
for i **in** $\texttt{S}$ **do**
  $(g, D) \leftarrow \text{Add}(g, D, i, w_i)$
end for
for i **in** $\texttt{S}$ **do**
  if $w_i = 1$ **then continue**; ▷ *We cannot swap i out if $\sum_{j\neq i} w_j = 0$!*
  ▷ *First zero out w_i.*
  $(g, D) \leftarrow \text{Add}(g, D, i, -w_i)$
  $(g, D) \leftarrow \text{Scale}(g, D, \frac{1}{1-w_i})$
  $w_i = 0$
  ▷ *Next perform line search to add back a point.*
  $\alpha = (D - g)./(D - 2g + \texttt{Kdiag})$; ▷ $\alpha_i = \arg\min_t \text{MMD}^2_{\boldsymbol{k}_{\mathbb{P}}}(\mathbb{S}_n^{te_i+(1-t)w}, \mathbb{P}) = \arg\min_t (1-t)^2 D + 2t(1-t)g + t^2 \texttt{Kdiag}$
  if $\texttt{fmt} = \text{SPLX}$ **then**
    $\alpha = \texttt{clip}(\alpha, 0, 1)$; ▷ *Clipping α to $[0,1]$. This corresponds to* $\arg\min_{t\in[0,1]} \text{MMD}^2_{\boldsymbol{k}_{\mathbb{P}}}(\mathbb{S}_n^{te_i+(1-t)w}, \mathbb{P})$.
  end if
  $D' \leftarrow (1-\alpha)^2 D + 2\alpha(1-\alpha)g + \alpha^2 \texttt{Kdiag}$ ▷ *multiplications are element-wise*
  $k \leftarrow \arg\min_i D'_i$
  $(g, D) \leftarrow \text{Scale}(g, D, 1 - \alpha_k)$
  $(g, D) \leftarrow \text{Add}(g, D, k, \alpha_k)$
end for
Return: $w \in \Delta_{n-1}$

Part II

Optimizing Continuous Distributions with Neural Parameterizations

In Part I, we saw optimizing over discrete distributions parameterized as weighted particles can lead to high-quality sampling. While particles are convenient to work and theoretical guarantees can be obtained for interpreting the optimization objective, for assessing the quality of the resulting samples, and for analyzing the algorithmic convergence, discrete representations of distributions have limitations. To list a few:

- For the generative modeling task, using particles alone cannot generate fresh samples unless more involved mechanism is used (e.g. Scarvelis, Borde, and Solomon [SBS23], whose generation time is linear in the dataset size).
- Quantities such as densities are hard to extract from particles and approximations such as kernel density estimation [Che17] need to be used.
- For high dimensions, it can take a large number of particles that is exponential in dimension to accurately represent the distribution, even if the distribution is simple such as a uniform measure in a box. For example, guarantees in Chapter 3 have exponents in the kernel growth rates which can scale exponentially in d (Prop. 3.1).

To tackle the above limitations, in Part II, we parameterize continuous distributions using neural networks in three different ways:

- as dual potentials to find the Wasserstein barycenter of continuous distributions (Chapter 4),
- as pushforward maps to locate minima of non-convex classical optimization (Chapter 5), and

- as time-varying velocity fields to solve mass-conserving partial differential equations (Chapter 6).

The central theme of Part II is to exploit *convexity principles* in deriving formulations amenable with neural networks with state-of-the-art performance in the respective task.

Chapter 4

Continuous Regularized Wasserstein Barycenters

In this chapter, we show how convex duality can be used to derive a Wasserstein barycenter formulation. This approach transitions from intractable optimization over distributions to tractable optimization over continuous functions. These functions can then be effectively parameterized as neural networks that are optimized using techinques from deep learning. This chapter is based on the publication Li et al. [Li+20].

4.1 Introduction

In statistics and machine learning, it is often desirable to aggregate distinct but similar collections of information, represented as probability distributions. For example, when temperature data is missing from one weather station, one can combine the temperature histograms from nearby stations to provide a good estimate for the missing station [Sol+14]. Or, in a Bayesian inference setting, when inference on the full data set is not allowed due to privacy or efficiency reasons, one can distributively gather posterior samples from slices of the data to form a single posterior incorporating information from all the data [Min+14; SLD18; Sri+15; Sta+17].

One successful aggregation strategy is to compute a *barycenter* of the input distributions. Given a notion of distance between distributions, the barycenter is the distribution that minimizes the sum of distances to the individual input distributions. A popular choice of distance is the *Wasserstein distance* based on the theory of optimal transport. The corresponding barycenter, called the *Wasserstein barycenter* was first studied in [AC11]. Intuitively, the Wasserstein distance is defined as the least amount of work required to transport the mass from one distribution into the other, where the notion of work is measured with respect to the metric of the underlying space on which the distributions are supported. The Wasserstein distance enjoys strong theoretical properties [Vil08; FG15; San15], and efficient algorithms for its computation have been proposed in recent

years [Cut13; Gen+16; Seg+17; PC19]. It has found success in many machine learning applications, including Bayesian inference [EM12] and domain adaptation [CFT14].

Finding the Wasserstein barycenter is not an easy task. To make it computationally tractable, the barycenter is typically constrained to be a discrete measure on a fixed number of support points [CD14; Sta+17; Dvu+18; CCS18]. This discrete approximation, however, can be undesirable in downstream applications, as it goes against the inherently continuous nature of many data distributions and lacks the capability of generating fresh samples when needed. To address this shortcoming, in this work we compute a continuous approximation of the barycenter that provides a stream of samples from the barycenter.

Contributions. We propose a stochastic optimization algorithm to approximate the Wasserstein barycenter without discretizing its support. Our method relies on a novel dual formulation of the regularized Wasserstein barycenter problem where the regularization is applied on a *continuous* support measure for the barycenter. The dual potentials that solve this dual problem can be used to recover the optimal transport plan between each input distribution and the barycenter. We solve the dual problem using stochastic gradient descent, yielding an efficient algorithm that only requires sample access to the input distributions. The barycenter can then be extracted as a follow-up step. Compared to existing methods, our algorithm produces the first continuous approximation of the barycenter that allows sample access. We demonstrate the effectiveness of our approach on synthesized examples and on real-world data for subset posterior aggregation.

■ 4.2 Related Works

In [AC11], the notion of Wasserstein barycenters was first introduced and analyzed theoretically. Although significant progress has been made in developing fast and scalable methods to compute the Wasserstein distance between distributions in both discrete [Cut13] and continuous cases [Gen+16; Seg+17], the search for an efficient and flexible Wasserstein barycenter algorithm has been overlooked in the continuous setting.

To have a tractable representation of the barycenter, previous methods assume that the barycenter is supported on discrete points. When the barycenter support is fixed *a priori*, the problem boils down to estimating the weights of the support points, and efficient projection-based methods can be used for discrete input measures [Ben+15; Sol+15; CP16] while gradient-based solvers can be used for continuous input measures [Sta+17; Dvu+18]. These fixed-support methods become prohibitive in higher dimensions, as the number of points required for a reasonable *a priori* discrete support grows exponentially. When the support points are free to move, alternating optimization of the support weights and the support points is typically used to deal with the non-convexity of the problem [CD14]. More recent methods use stochastic optimization [CCS18] or the

Franke–Wolfe algorithm [Lui+19] to construct the support iteratively. These free-support methods, however, are computationally expensive and do not scale to a large number of support points.

If the support is no longer constrained to be discrete, a key challenge is to find a suitable representation of the now continuous barycenter, a challenge that is unaddressed in previous work. We draw inspiration from [Gen+16], where the Wasserstein distance between continuous distributions is computed by parameterizing the dual potentials in a reproducing kernel Hilbert space (RKHS). Their work was followed by [Seg+17], where neural networks are used instead of RKHS parameterizations. The primal-dual relationship exhibits a bridge between continuous dual potentials and the transport plans, which can the be marginalized to get a convenient continuous representation of the distributions. However, a direct extension of [Seg+17] to the barycenter problem will need to parameterize the barycenter measure, resulting in an alternating min-max optimization scheme. By introducing a novel regularizing measure that does not rely on the unknown barycenter but only on a proxy support measure, we are able to encode the information of the barycenter in the dual potentials themselves without explicitly parameterizing the barycenter. This idea of computing the barycenter from dual potentials can be viewed as a generalization of [CP16] to the continuous case where the barycenter is no longer supported on a finite set known beforehand.

■ 4.3 Background on Optimal Transport

Throughout, we consider a compact set $\mathcal{X} \subset \mathbb{R}^d$ equipped with a symmetric continuous cost function $c : \mathcal{X} \times \mathcal{X} \to \mathbb{R}_{\geq 0}$. We denote by $\mathcal{P}(\mathcal{X})$ the space of probability measures on $\mathcal{X}$. For any $\mu, \nu \in \mathcal{P}(\mathcal{X})$, the Kantorovich formulation of optimal transport between μ and ν is defined as:

$$W(\mu, \nu) \triangleq \inf_{\pi \in \Pi(\mu,\nu)} \int c(x, y) \, \mathrm{d}\pi(x, y), \tag{4.1}$$

where $\Pi(\mu, \nu) \triangleq \{\pi \in \mathcal{P}(\mathcal{X}^2) : (P_x)_{\#}\pi = \mu, (P_y)_{\#}\pi = \nu\}$ is the set of admissable *transport plans*, $P_x(x, y) \triangleq x$ and $P_y(x, y) \triangleq y$ are the projections onto the first and second coordinate respectively, and $T_{\#}\alpha$ denotes the pushforward of the measure α by a function T. When $c(x, y) = ||x - y||_2^p$, the quantity $W(\mu, \nu)^{1/p}$ is the *p-Wasserstein* distance between μ and ν.

The primal problem (4.1) admits an equivalent dual formulation [San15, Theorem 1.42]:

$$W(\mu, \nu) = \sup_{\substack{f,g \in C(\mathcal{X}) \\ f \oplus g \leq c}} \int_{\mathcal{X}} f(x) \, \mathrm{d}\mu(x) + \int_{\mathcal{X}} g(y) \, \mathrm{d}\nu(y), \tag{4.2}$$

where $C(\mathcal{X})$ is the space of continuous real-valued functions on $\mathcal{X}$, and $(f \oplus g)(x, y) \triangleq$

$f(x) + g(y)$. The inequality $f \oplus g \leq c$ is interpreted as $f(x) + g(y) \leq c(x, y)$ for μ-a.e. x and ν-a.e. y. We refer to f and g as the *dual potentials*.

Directly solving (4.1) and (4.2) is challenging: even with discretization, the resulting linear program can be large. Hence *regularized* optimal transport has emerged as a popular, efficient alternative [Cut13]. Let $\xi \in \mathcal{P}(\mathcal{X}^2)$ be the measure on which we enforce a relaxed version of the constraint $f \oplus g \leq c$. We call ξ the *regularizing measure*. In previous work, ξ is usually taken to be the product measure $\mu \otimes \nu$ [Gen+16] or the uniform measure on a discrete set of points [Cut13]. Given a convex function $R : \mathbb{R} \to \mathbb{R}$, we define the regularized version of (4.1) with respect to ξ, R as

$$W_R^\xi(\mu, \nu) \triangleq \inf_{\substack{\pi \in \Pi(\mu,\nu) \\ \pi \ll \xi}} \int_{\mathcal{X}\times\mathcal{X}} c(x, y)\, \mathrm{d}\pi(x, y) + \int_{\mathcal{X}\times\mathcal{X}} R\left(\frac{\mathrm{d}\pi}{\mathrm{d}\xi}(x, y)\right) \mathrm{d}\xi(x, y), \tag{4.3}$$

where $\pi \ll \xi$ denotes that π is absolutely continuous with respect to ξ and $\frac{\mathrm{d}\pi}{\mathrm{d}\xi}$ denotes the Radon-Nikodym derivative. In this work, we consider entropic and quadratic regularization defined by

$$\forall t \geq 0, \quad R(t) \triangleq \begin{cases} \epsilon(t \ln t - t) & \text{entropic} \\ \frac{\epsilon}{2} t^2 & \text{quadratic.} \end{cases} \tag{4.4}$$

As in the unregularized case, the primal problem (4.3) admits an equivalent dual formulation for entropic [Gen+16; Cla+19] and quadratic [LMM19] regularization:

$$W_R^\xi(\mu, \nu) = \sup_{f,g \in C(\mathcal{X})} \int_{\mathcal{X}} f(x)\, \mathrm{d}\mu(x) + \int_{\mathcal{X}} g(y)\, \mathrm{d}\nu(y) - \int_{\mathcal{X}\times\mathcal{X}} R^*\left(f(x) + g(y) - c(x, y)\right) \mathrm{d}\xi(x, y), \tag{4.5}$$

where the function R^* on the dual problem is determined as

$$\forall t \in \mathbb{R}, \quad R^*(t) = \begin{cases} \epsilon \exp(\frac{t}{\epsilon}) & \text{entropic} \\ \frac{1}{2\epsilon}(t_+)^2 & \text{quadratic.} \end{cases}$$

The regularized dual problem has the advantage of being *unconstrained* thanks to the penalization of R^* to smoothly enforce $f \oplus g \leq c$. We can recover the optimal transport plan π that solves (4.3) from the optimal dual potentials (f, g) that solves (4.5) using the primal-dual relationship [Gen+16; LMM19]:

$$\mathrm{d}\pi(x, y) = H(x, y)\mathrm{d}\xi(x, y), \text{ where } H(x, y) = \begin{cases} \exp\left(\frac{f(x)+g(y)-c(x,y)}{\epsilon}\right) & \text{entropic} \\ \left(\frac{f(x)+g(y)-c(x,y)}{\epsilon}\right)_+ & \text{quadratic.} \end{cases} \tag{4.6}$$

Entropic regularization is more popula because in the discrete case it yields a problem that can be solved with the celebrated Sinkhorn algorithm [Cut13]. We will consider both entropic and quadratic regularization in our experimental setup, while the method developed in the next section is applicable to more general regularization.

■ 4.4 Regularized Wasserstein Barycenters

We now use the regularized Wasserstein distance (4.3) to define a regularized version of the Wasserstein barycenter problem introduced in [AC11].

◇ 4.4.1 Primal and dual formulations

Given input distributions $\mu_1, \dots, \mu_n \in \mathcal{P}(\mathcal{X})$ and weights $\lambda_1, \dots, \lambda_n \in \mathbb{R}_{\geq 0}$, the *(unregularized) Wasserstein barycenter* problem[1] of these input measures with respect to the weights is [AC11]:

$$\inf_{\nu \in \mathcal{P}(\mathcal{X})} \sum_{i=1}^{n} \lambda_i W(\mu_i, \nu). \tag{4.7}$$

Since this formulation is hard to solve in practice, we instead consider the following *regularized Wasserstein barycenter* problem with respect to the regularized Wasserstein distance (4.3) and some $\eta \in \mathcal{P}(\mathcal{X})$:

$$\inf_{\nu \in \mathcal{P}(\mathcal{X})} \sum_{i=1}^{n} \lambda_i W_R^{\mu_i \otimes \eta}(\mu_i, \nu). \tag{4.8}$$

If we knew the true barycenter ν, one clear choice is $\eta = \nu$. For (4.8) to make sense without referring to ν, we must use another measure η as a proxy for ν. We call such η the *barycenter support measure*. If no information about the barycenter is known beforehand, we take $\eta = \text{Uniform}(\mathcal{X})$, the uniform measure on $\mathcal{X}$. Otherwise we can choose η based on the information we have.

Our method relies on the following dual formulation of (4.8):

Theorem 4.4.1. *Let $\mathcal{X} \subset \mathbb{R}^d$ be a compact domain. Consider input distributions $\mu_1, \dots, \mu_n \in \mathcal{P}(\mathcal{X})$, weights $\lambda_1, \dots, \lambda_n \in \mathbb{R}_{\geq 0}$, a support measure $\eta \in \mathcal{P}(\mathcal{X})$, and regularizing functions R, R^* so that W_R^ξ admits primal and dual formulations (4.3) and (4.5) for any $\xi \in \mathcal{P}(\mathcal{X}^2)$. Suppose in addition that R^* is convex and increasing. Then the dual problem of (4.8) is*

$$\sup_{\substack{\{(f_i, g_i)\}_{i=1}^n \subset C(\mathcal{X})^2 \\ \sum_{i=1}^n \lambda_i g_i = 0}} \sum_{i=1}^{n} \lambda_i \left(\int f_i \, \mathrm{d}\mu_i - \iint R^* \left(f_i(x) + g_i(y) - c(x,y) \right) \mathrm{d}\mu_i(x) \, \mathrm{d}\eta(y) \right). \tag{4.9}$$

The (strong) duality holds in the sense that the infimum of (4.8) equals the supremum of (4.9), and a solution to (4.8) exists. If $\{(f_i, g_i)\}_{i=1}^n$ solves (4.9), then each (f_i, g_i) is a solution to the dual formulation (4.5) of $W_R^{\mu_i \otimes \eta}(\mu_i, \nu)$, where ν is the barycenter solution of (4.8).

[1] In some convention, there is an additional exponent in the summands of (4.7) such as $\sum_{i=1}^n \lambda_i W_2^2(\mu_i, \nu)$ for the 2-Wasserstein barycenter. Here we absorb such exponent in (4.1), e.g., $W(\mu_i, \nu) = W_2^2(\mu_i, \nu)$.

The proof given below relies on the convex duality theory of locally convex topological spaces developed in [ET99].

Remark 4.1. *Based on Thm. 4.4.1, we can recover the optimal transport plan π_i between μ_i and the barycenter ν from the pair (f_i, g_i) that solves (4.9) via the primal-dual relationship (4.6).*

Proof of Thm. 4.4.1. We first prove the strong duality. Recall that $\mathcal{X} \subset \mathbb{R}^d$ is compact. We thus view $C(\mathcal{X})$ as a normed vector space with the supremum norm. Let $V \triangleq ((C(\mathcal{X}))^2)^n$ be the direct sum vector space endowed with the natural norm, i.e., for $u = \{(f_i, g_i)\}_{i=1}^n \in V$,

$$\|u\| \triangleq \sum_{i=1}^n (\|f_i\| + \|g_i\|).$$

For brevity, denote $\xi_i \triangleq \mu_i \otimes \eta$. We use the notation ξ_i to suggest that a more general support measure can be used to establish the strong duality. Define $J : V \to \mathbb{R}$ to be, for $u = \{(f_i, g_i)\}_{i=1}^n$,

$$\begin{aligned} J(u) &\triangleq \sum_{i=1}^n \lambda_i \left(\iint R^*(f_i(x) + g_i(y) - c(x,y)) \mathrm{d}\xi_i(x,y) - \int f_i(x) \mathrm{d}\mu_i(x) \right) \\ &= \sum_{i=1}^n \lambda_i \left(\iint R^*(f_i \oplus g_i - c) \mathrm{d}\xi_i - \int f_i \mathrm{d}\mu_i \right). \end{aligned} \tag{4.10}$$

Let $Y \triangleq C(\mathcal{X})$. By the Riesz–Markov–Kakutani representation theorem, the continuous dual space of Y is $Y^* = \mathcal{M}(\mathcal{X})$, the space of regular Borel measures. Define $B : V \to Y$ as, for $u = \{(f_i, g_i)\}_{i=1}^n$,

$$B(u) = B(\{(f_i, g_i)\}_{i=1}^n) \triangleq -\sum_{i=1}^n \lambda_i g_i.$$

Then the optimization (4.9) becomes, after negating the objective,

$$\inf_{\substack{u \in V \\ B(u) = 0}} J(u),$$

where the equality $B(u) = 0$ is component-wise (i.e. $B(u)$ is the constant-zero function in Y). Similarly we use $\leq, <$ to mean component-wise inequalities in $C(\mathcal{X})$. We claim the above program is the same as

$$\inf_{\substack{u \in V \\ B(u) \leq 0}} J(u). \tag{4.11}$$

This is because if $u \in \{u : B(u) \le 0\}$, then for any $x \in \mathcal{X}$, $\sum_{i=1}^n \lambda_i g_i(x) \ge 0$, and by replacing every g_i with $g_i - \sum_{i=1}^n \lambda_i g_i(x)$ the objective (4.11) can only gets smaller due to the monotonicity of R^*.

The dual problem of (4.11) can be calculated as (see Chapter III.(5.23) in [ET99])

$$
\begin{aligned}
&\sup_{\nu \le 0} \inf_{u \in V} \left\{ -\int B(u) \mathrm{d}\nu + J(u) \right\} \\
=&\sup_{\nu \le 0} \inf_{\{(f_i,g_i)\}_{i=1}^n \subset C(\mathcal{X})^2} -\int \left(-\sum_{i=1}^n \lambda_i g_i \right) \mathrm{d}\nu + \sum_{i=1}^n \lambda_i \left(\iint R^*(f_i \oplus g_i - c) \mathrm{d}\xi_i - \int f_i \mathrm{d}\mu_i \right) \\
=&\sup_{\nu \ge 0} \inf_{\{(f_i,g_i)\}_{i=1}^n \subset C(\mathcal{X})^2} \sum_{i=1}^n \lambda_i \left(\iint R^*(f_i \oplus g_i - c) \mathrm{d}\xi_i - \int f_i \mathrm{d}\mu_i - \int g_i \mathrm{d}\nu \right) \\
=&\sup_{\nu \ge 0} \sum_{i=1}^n \lambda_i \inf_{(f_i,g_i) \in C(\mathcal{X})^2} \left(\iint R^*(f_i \oplus g_i - c) \mathrm{d}\xi_i - \int f_i \mathrm{d}\mu_i - \int g_i \mathrm{d}\nu \right) \qquad (4.12) \\
=&\sup_{\nu \ge 0} \sum_{i=1}^n -\lambda_i W_R^{\xi_i}(\mu_i, \nu) \qquad (4.13) \\
=&-\inf_{\nu \ge 0} \sum_{i=1}^n \lambda_i W_R^{\xi_i}(\mu_i, \nu). \qquad (4.14)
\end{aligned}
$$

To get (4.13) we used the duality for regularized Wasserstein distance (4.5).

In order to apply classical results from convex analysis (for instance, Proposition 5.1 of Chapter III in [ET99]) to establish the strong duality and the existence of solutions, we need to show:

(a) J is a convex l.s.c. (lower-semicontinuous) function.

(b) B is convex.

(c) For any $\nu \in Y^*$, $\nu \ge 0$, the map $u \mapsto \int B(u) \mathrm{d}\nu$ is l.s.c.

(d) $\{u \in V : B(u) \le 0\} \ne \varnothing$.

(e) There exists $u_0 \in V$ such that $-B(u_0) < 0$.

(f) The infimum in (4.11) is finite.

Since B is linear and $Y = C(\mathcal{X})$ in our case, the conditions (b)-(e) are satisfied automatically. Convexity of J (a) follows because R^* is convex so that, for $u_j = \{(f_i^{(j)}, g_i^{(j)})\}_{i=1}^n$,

$j \in \{1,2\}$, and $\theta \in [0,1]$,

$$\begin{aligned}
&J(\theta u_1 + (1-\theta)u_2) \\
&= \sum_{i=1}^n \lambda_i \left(\iint R^*((\theta f_i^{(1)} + (1-\theta) f_i^{(2)}) \oplus \theta(g_i^{(1)} + (1-\theta) g_i^{(2)})) - c) \mathrm{d}\xi_i \right. \\
&\qquad \left. - \int (\theta f_i^{(1)} + (1-\theta) f_i^{(2)}) \mathrm{d}\mu_i \right) \\
&\le \theta \sum_{i=1}^n \lambda_i \left(\iint R^*(f_i^{(1)} \oplus g_i^{(1)} - c) \mathrm{d}\xi_i - \int f_i^{(1)} \mathrm{d}\mu_i \right) \\
&\qquad + (1-\theta) \sum_{i=1}^n \lambda_i \left(\iint R^*(f_i^{(2)} + g_i^{(2)} - c) \mathrm{d}\xi_i - \int f_i^{(2)} \mathrm{d}\mu_i \right) \\
&= \theta J(u_1) + (1-\theta) J(u_2).
\end{aligned}$$

Next we show that J is l.s.c. with respect to the norm topology on V. Since J is convex and does not take on values $\pm\infty$, by Proposition III.2.5 of [ET99], it is enough to show that J is bounded above in a neighborhood of 0. Fix any $\delta > 0$. As before we write $u = \{(f_i, g_i)\}_{i=1}^n \in V$. Then $\|u\| < \delta$ implies $\sup_{x \in \mathcal{X}} \max(f_i(x), g_i(x)) < \delta$ for all i. Since $\mathcal{X}$ is compact, $\sup_{x,y \in \mathcal{X}} c(x,y)$ is bounded. Hence the integrand in (4.10) is bounded for $\|u\| < \delta$ as R^* is increasing, and the conclusion that J is bounded on $\{u \in V : \|u\| < \delta\}$ follows from the fact that both ξ_i and μ_i are probability measures for all i. This proves J is continuous, and in particular l.s.c.

It remains to show that the infimum in (4.11) is finite. Note that for $u \in V$ such that $B(u) \le 0$, we have $\sum_{i=1}^n \lambda_i g_i \ge 0$. Hence in this case, if let λ be the uniform measure on $\mathcal{X}$, then

$$\begin{aligned}
J(u) &= \sum_{i=1}^n \lambda_i \left(\iint R^*(f_i \oplus g_i - c) \mathrm{d}\xi_i - \int f_i \mathrm{d}\mu_i \right) \\
&\ge \sum_{i=1}^n \lambda_i \left(\iint R^*(f_i \oplus g_i - c) \mathrm{d}\xi_i - \int f_i \mathrm{d}\mu_i \right) - \int \left(\sum_{i=1}^n \lambda_i g_i \right) \mathrm{d}\lambda \\
&= \sum_{i=1}^n \lambda_i \left(\iint R^*(f_i \oplus g_i - c) \mathrm{d}\xi_i - \int f_i \mathrm{d}\mu_i - \int g_i \mathrm{d}\lambda \right) \\
&\ge -\sum_{i=1}^n \lambda_i W_R^{\mu_i \otimes \lambda}(\mu_i, \lambda) \\
&> -\infty.
\end{aligned}$$

Thus by Proposition III.5.1 of [ET99], the problem (4.11) is stable (Definition III.2.2 in [ET99]), and in particular normal, so we have the strong duality (Proposition III.2.1, III.2.2 in [ET99]), and the dual problem (4.14) has at least one solution. We comment that this does not imply (4.11) has a solution.

To show the solution ν^* to (4.14) is actually a probability measure, suppose $\nu^*(\mathcal{X}) \neq 1$. Consider the inner infimum in (4.12) for a particular i. For any $t \in \mathbb{R}$, we can set $f = t$ and $g = -t$. Then

$$\begin{aligned}
&\int R^*\left(f \oplus g - c\right) \mathrm{d}\xi_i - \int f \mathrm{d}\mu_i - \int g \mathrm{d}\nu^* \\
&= \int R^*\left(-c\right) \mathrm{d}\xi_i + t(\nu^*(\mathcal{X}) - \mu_i(\mathcal{X})) \\
&\leq R^*(0) + t(\nu^*(\mathcal{X}) - \mu_i(\mathcal{X})),
\end{aligned}$$

where we used the fact that R^* is increasing and $c \geq 0$. Either sending $t \to \infty$ or $t \to -\infty$ shows that the minimizer ν^* must satisfy $\nu^*(\mathcal{X}) = \mu_i(\mathcal{X}) = 1$, for otherwise the infimum would be $-\infty$, which contradicts the strong duality and (f).

Finally we prove the last statement of Thm. 4.4.1. That is, if $\{(f_i, g_i)\}_{i=1}^n$ solves (4.9), then each pair (f_i, g_i) solves (4.5). Suppose that $\{(f_i, g_i)\}_{i=1}^n$ solves (4.9). Let ν^* denote the solution to (4.8). Then $\sum_{i=1}^n \lambda_i g_i = 0$. Hence the supremum of (4.9) equals

$$\begin{aligned}
&\sum_{i=1}^n \lambda_i \left(\int f_i \mathrm{d}\mu_i + \int g_i \mathrm{d}\nu^* - \iint R^*(f_i \oplus g_i - c) \mathrm{d}\mu_i \mathrm{d}\eta \right) \\
&\leq \sum_{i=1}^n \lambda_i W_R^{\mu_i \otimes \eta}(\mu_i, \nu^*),
\end{aligned} \tag{4.15}$$

where the inequality follows from the duality (4.5) of the regularized Wasserstein distance. By the strong duality we just showed, the supremum of (4.9) equals the infimum (4.8) which is (4.15). Hence the inequality in (4.15) is an equality, and we see that each pair (f_i, g_i) solves (4.5). □

♢ 4.4.2 Solving the regularized barycenter problem

Notice that (4.9) is convex in the potentials $\{f_i, g_i\}_{i=1}^n$ with the linear constraint $\sum_{i=1}^n \lambda_i g_i = 0$. To get an unconstrained version of the problem, we replace each g_i with $g_i - \sum_{i=1}^n \lambda_i g_i$. Rewriting integrals as expectations, we obtain the following formulation equivalent to (4.9):

$$\sup_{\substack{\{f_i\}_{i=1}^n \subset C(\mathcal{X}) \\ \{g_i\}_{i=1}^n \subset C(\mathcal{X})}} \mathbb{E}_{\substack{X_i \sim \mu_i \\ Y \sim \eta}} \left[\sum_{i=1}^n \lambda_i \left(f_i(X_i) - R^* \left(f_i(X_i) + g_i(Y) - \sum_{j=1}^n \lambda_j g_j(Y) - c(X_i, Y) \right) \right) \right]. \tag{4.16}$$

This new formulation is an unconstrained concave maximization.

The optimization space of (4.16) is infinite-dimensional. Following [Gen+16], we parameterize the potentials $\{f_i, g_i\}_{i=1}^n$ and solve (4.16) using stochastic gradient descent. We summarize our algorithm in Alg. 4.4.1. In their paper, the parameterization is done using

reproducing kernel Hilbert spaces, which can be made more efficient using random Fourier features [RR08]; this technique gives convergence guarantees but is only well-suited for smooth problems. In [Seg+17], a neural network parameterization is used with the benefit of approximating arbitrary continuous functions, but its convergence guarantees are more elusive. We extend these techniques to solve (4.16). A comparison between neural network parameterization and random Fourier parameterization is included in Fig. 2.

Once we approximate the optimal potentials $\{f_i\}_{i=1}^n, \{g_i\}_{i=1}^n$ in (4.16), as observed in Rem. 4.1, we can recover the corresponding transport plan π_i via the primal-dual relationships (4.6).

This formulation can be easily extended to the discrete case. If the barycenter has a fixed discrete support known *a priori*, we take η to be the uniform measure on the discrete support and parameterize each g_i as a real-valued vector. If the input distributions are discrete, we can use an analogous discrete representation for each f_i.

Algorithm 4.4.1 Stochastic gradient descent to solve the regularized barycenter problem (4.16)

Input: distributions $\mu_1, \dots, \mu_n$ with sample access, weights $(\lambda_1, \dots, \lambda_n)$, dual regularizer R^*, regularizing measure η, cost function c, gradient update function `GradientUpdate`
Initialize parameterizations $\{(f_{\theta_i}, g_{\phi_i})\}_{i=1}^n$
for $l \leftarrow 1$ **to** n_{epochs} **do**
  for $i \leftarrow 1$ **to** n **do**
    Sample $x^{(i)} \sim \mu_i$ and $y \sim \eta$
  end for
  $\bar{g} \leftarrow \sum_{i=1}^n \lambda_i g_{\phi_i}(y)$
  $F \leftarrow \sum_{i=1}^n \lambda_i \left(f_{\theta_i}(x^{(i)}) - R^* \left(f_{\theta_i}(x^{(i)}) + g_{\phi_i}(y) - \bar{g} - c(x^{(i)}, y) \right) \right)$
  for $i \leftarrow 1$ **to** n **do**
    $\theta_i \leftarrow \texttt{GradientUpdate}(\theta_i, -\nabla_{\theta_i} F)$
    $\phi_i \leftarrow \texttt{GradientUpdate}(\phi_i, -\nabla_{\phi_i} F)$
  end for
end for
Return: dual potentials $\{(f_{\theta_i}, g_{\phi_i})\}_{i=1}^n$

♢ 4.4.3 Recovering the barycenter

By Thm. 4.4.1, for any i, once we solve the optimal potentials and recover the transport plan $\pi_i \in \mathcal{P}(\mathcal{X}^2)$ using (4.6), the barycenter ν equals $(P_y)_{\#}\pi_i$. While this pushforward is straightforward to evaluate when π_i's are discrete, in the continuous setting such marginal-

ization is difficult, especially when the dimension of $\mathcal{X}$ is large. Below we consider a few ways to recover the barycenter from the transport plans:

(a) Use numerical integration to approximate $(P_y)_\# \pi_i(x) = \int \pi_i(x, y)\,\mathrm{d}y$ with proper discretization of the space $\mathcal{X}$, if π_i has density (if the input distributions and η have densities by (4.6)).

(b) Use Markov chain Monte Carlo (MCMC) methods to sample according to π_i, again assuming it has (unnormalized) density, and then take the second components of all the samples.

Option (a) is only viable for small dimensions. Option (b) is capable of providing quality samples, but is slow in practice and requires case-by-case parameter tuning. Both (a) and (b) additionally require knowing the densities of input distributions to evaluate π_i, which may not be available in practice.

A different kind of approach is to estimate a *Monge map* approximating each π_i. Formally, a Monge map from $\mu \in \mathcal{P}(\mathcal{X})$ to $\nu \in \mathcal{P}(\mathcal{X})$ is a solution to

$$\inf_{T:\mathcal{X}\to\mathcal{X}, T_\#\mu=\nu} \int_{\mathcal{X}} c(x, T(x))\mathrm{d}\mu(x).$$

When the cost satisfies $c(x, y) = h(x - y)$ with a convex h and μ has density, it is linked to the optimal transport plan π between μ and ν by $\pi = (\mathrm{id}, T)_\#\mu$ [San15]. With regularization, such exact correspondence may not hold. Nevertheless π encodes the crucial information of a Monge map when the regularization is small. If we can find $T_i : \mathcal{X} \to \mathcal{X}$ that realizes π_i for each i, then we can recover the barycenter as $\sum_{i=1}^n \lambda_i (T_i)_\#\mu_i$. In the unregularized case, all of $(T_i)_\#\mu_i$ should agree. In practice, we have found that taking the weighted average of $(T_i)_\#\mu_i$'s helps reduce the error brought by each individual T_i. We consider the following variants of Monge map estimation:

(c) Compute pointwise barycentric projection [CFT14; Seg+17]. If $c(x, y) = \|x - y\|_2^2$, then barycentric projection takes the simplified form

$$T_i(x) = \mathbb{E}_{Y\sim\pi_i(\cdot:x)}[Y]. \tag{4.17}$$

(d) Recover an approximation of the Monge map using the gradient of the dual potentials [TJ19]. For the case when $c(x, y) = \|x - y\|_2^2$ and the densities of the source distributions exist, there exists a unique Monge map realizing the (unregularized) optimal transport plan π_i [San15]:

$$T_i(x) = x - \frac{1}{2}\nabla f_i(x).$$

While this does not strictly hold for the regularized case, it gives a cheap approximation of T_i's.

(e) Find T_i as a solution to the following optimization problem [Seg+17], where H is defined in (4.6):

$$T_i \triangleq \operatorname*{arg\,min}_{T:\mathcal{X}\to\mathcal{X}} \mathbb{E}_{(X,Y)\sim\pi_i}\left[c(T(X),Y)\right] = \operatorname*{arg\,min}_{T:\mathcal{X}\to\mathcal{X}} \mathbb{E}_{\substack{X\sim\mu_i \\ Y\sim\eta}}\left[c(T(X),Y)H(X,Y)\right]. \quad (4.18)$$

In [Seg+17] each T_i is parameterized as a neural network. In practice, the regularity of the neural networks smooths the transport map, avoiding erroneous oscillations due to sampling error in methods like barycentric projection (c) where each T_i is estimated pointwise.

Compared to (a)(b), options (c)(d)(e) do no require knowing the densities of the input distributions. See a comparison of these methods in Fig. 1.

■ 4.5 Implementation and Experiments

We tested the proposed framework for computing a continuous approximation of the barycenter on both synthetic and real-world data. In all experiments we use equal weights for input distributions, i.e., $\lambda_i = \frac{1}{n}$ for all $i = 1, \ldots, n$. Throughout we use the squared Euclidean distance as the cost function, i.e., $c(x, y) = \|x - y\|_2^2$. Note that our method is not limited to Euclidean distance costs and can be generalized to different cost functions in $\mathbb{R}^d$—even to distance functions on curved domains. The source code is publicly available at https://github.com/lingxiaoli94/CWB.

Implementation details. The support measure η is set to be the uniform measure on a box containing the support of all the source distributions, estimated by sampling.

For $c(x, y) = \|x - y\|_2^2$, we can simplify the (unregularized) Wasserstein barycenter problem by considering centered input distributions [Álv+16]. Concretely, if the mean of μ_i is m_i, then the mean of the resulting barycenter is $\sum_{i=1}^n \lambda_i m_i$, and we can first compute the barycenter of input distributions centered at 0 and then translate the barycenter to have the right mean. We adopt this simplification since this allows us to reduce the size of the support measure η when the input distributions are far apart. When computing the Monge map (c)(d)(e), for each i, we further force $(T_i)_{\#}\mu_i$ to have zero mean by replacing T_i with $T_i - \mathbb{E}_{X\sim\mu_i}[T_i(X)]$. We have found that empirically this helps reduce the bias coming from regularization when recovering the Monge map.

The stochastic gradient descent used to solve (4.16) and (4.18) is implemented in Tensorflow 2.1 [Aba+16]. In all experiments below, we use Adam optimizer [KB14] with learning rate 10^{-4} and batch size 4096 or 8192 for the training. The dual potentials $\{f_i, g_i\}_{i=1}^n$ in (4.16) are each parameterized as neural networks with two fully-connected layers ($d \to 128 \to 256 \to 1$) using the ReLU activation. Every T_i in (4.18) is parameterized with layers ($d \to 128 \to 256 \to d$). We have tested with deeper/wider network

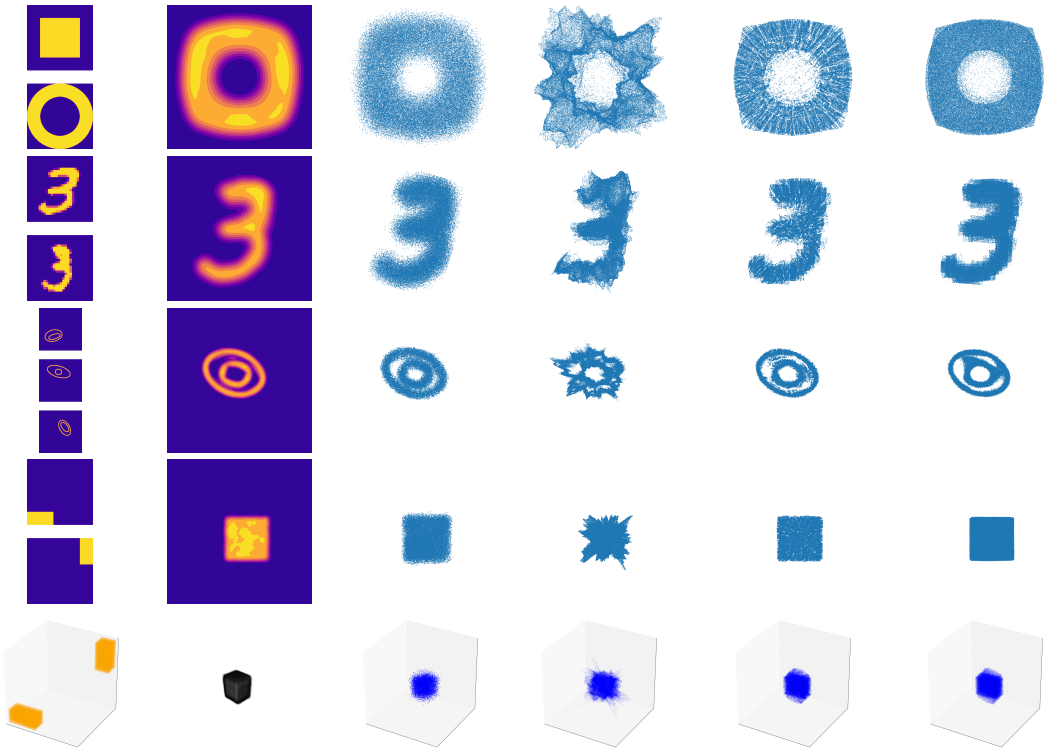

Figure 1: Comparison of barycenter recovery methods.

architectures but have found no noticeable improvement. We change the choice of the regularizer and the number of training iterations depending on the examples.

Qualitative results in 2 and 3 dimensions. Fig. 1 shows the results for methods (a)-(e) from Sec. 4.4.3 on various examples. For each example represented as a row, we first train the dual potentials using quadratic regularization with $\epsilon = 10^{-4}$ or $\epsilon = 10^{-5}$. Then each method is run subsequently to obtain the barycenter. Alg. 4.4.1 takes less than 10 minutes to finish for these experiments.[2] For (a) we use a discretized grid with grid size 200 in 2D and grid size 80 in 3D. For (b) we use Metropolis-Hastings to generate 10^5 samples with a symmetric Gaussian proposal. The results from (a)(b) are aggregated from all transport plans. For (c)(d)(e) we sample from each input distribution and then push the samples forward using T_i's to have 10^5 samples in total.

In short: (a) numerical integration shows the transport plans π_i's computed by (4.6) are accurate and smooth; (b) MCMC samples match the barycenter in (a) but are expen-

[2]We ran our experiments using a NVIDIA Tesla V100 GPU on a Google cloud instance with 12 compute-optimized CPUs and 64GB memory.

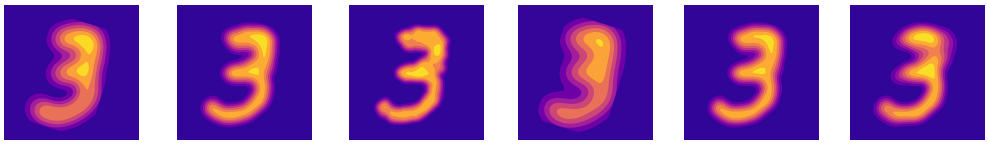

Figure 2: Comparison of regularization and parameterization choices. Labels at the bottom row are the regularizer type and the value of the constant ϵ as in (4.4). R_{L_2}, R_e means using quadratic and entropic regularization respectively. The last two columns show the result of using random Fourier features [RR08] instead of neural networks, with f indicating the scale of the frequencies used.

sive to compute and can be blurry near the boundaries; (c) barycentric projection yields poor boundaries due to the high variance in evaluating (4.17) pointwise; (d) gradient-based map has fragmented white lines in the interior; (e) the method by [Seg+17] can inherit undesirable artifact from the input distributions—for instance, in the last column of the second row the digit 3 looks pixelated.

Next, we compare the impact of the choice of regularization and parameterization in Fig. 2. We use the digit 3 example (row 2 in Fig. 1) and run numerical integration (a) to recover the barycenter. The first three columns confirm that smaller ϵ gives sharper results as the computed barycenter tends to the unregularized barycenter. On the other hand, entropic regularization yields a smoother marginal, but smaller ϵ leads to numerical instability: we display the smallest one we could reach. The last two columns show that parameterization using random Fourier features [RR08] gives a comparable result as using neural networks, but the scale of the frequencies needs to be fine-tuned.

Multivariate Gaussians with varying dimensions. When the input distributions are multivariate Gaussians, the (unregularized) barycenter is also a multivariate Gaussian, and an efficient fixed-point algorithm can be used to recover its parameters [Álv+16]. We compute the ground truth barycenter of 5 randomly generated multivariate Gaussians in varying dimensions using [Álv+16] and compare our proposed algorithm to other state-of-the-art barycenter algorithms. Since measuring the Wasserstein distance of two distributions in high dimensions is computationally challenging, we instead compare the maximum likelihood estimation (MLE) parameters if we fit a Gaussian to the computed barycenter samples and compare with the true parameters. See Tab. 1 for the results of our algorithm with quadratic regularization compared with those from other state-of-the-art free-support methods. Among the Monge map estimation methods, the gradient-based Monge map (d) works the best in higher dimensions, and the result of (e) is slightly worse: we believe this is due to the error accumulated in the second stochastic optimization used to compute (4.18). For brevity, we only include (d) in Tab. 1. Note that discrete fixed-

Table 1: Comparison of free-support barycenter algorithms on multivariate Gaussians of varying dimensions. Reported are the covariance difference $\|\Sigma - \Sigma^*\|_F$ where Σ is the MLE covariance of the barycenter computed by each method, Σ^* is the ground truth covariance, and $\|\cdot\|_F$ is the Frobenius norm. Smaller is better. All experiments are repeated 5 times with the mean and standard deviation reported. We use 5000 and 100 support points in [CD14] and [CCS18] respectively, as these are the maximum numbers allowed for the algorithms to terminate in a reasonable amount of time.

Dimension	[CD14]	[CCS18]	Ours with (d) and R_{L_2}
2	$\mathbf{7.28\times10^{-4}(9.99\times10^{-5})}$	$2.39\times10^{-3}(3.14\times10^{-4})$	$1.98\times10^{-3}(1.17\times10^{-4})$
3	$\mathbf{4.96\times10^{-3}(6.42\times10^{-4})}$	$8.97\times10^{-3}(9.22\times10^{-4})$	$5.05\times10^{-3}(6.32\times10^{-4})$
4	$1.35\times10^{-2}(1.73\times10^{-3})$	$2.50\times10^{-2}(1.68\times10^{-3})$	$\mathbf{1.22\times10^{-2}(1.44\times10^{-3})}$
5	$2.43\times10^{-2}(1.87\times10^{-3})$	$5.05\times10^{-2}(2.22\times10^{-3})$	$\mathbf{1.52\times10^{-2}(1.18\times10^{-3})}$
6	$4.38\times10^{-2}(2.04\times10^{-3})$	$8.86\times10^{-2}(2.58\times10^{-3})$	$\mathbf{2.37\times10^{-2}(3.24\times10^{-3})}$
7	$5.91\times10^{-2}(1.26\times10^{-3})$	$1.24\times10^{-1}(1.63\times10^{-3})$	$\mathbf{4.07\times10^{-2}(2.65\times10^{-3})}$
8	$8.31\times10^{-2}(1.23\times10^{-3})$	$1.64\times10^{-1}(1.48\times10^{-3})$	$\mathbf{4.23\times10^{-2}(3.14\times10^{-3})}$

support algorithms will have trouble scaling to higher dimensions as the total number of grid points grows exponentially with the number of dimensions. For instance, the covariance difference between the ground truth and those from running [Sta+17] with 10^5 support points in $\mathbb{R}^4$ is $5.99 \times 10^{-2}(\pm 6.19 \times 10^{-3})$, which is significantly worse than the ones shown in Tab. 1. See Sec. 4.A.1 for more details. In this experiment, we are able to consistently outperform state-of-the-art free-support methods in higher dimensions with the additional benefit of providing sample access from the barycenter.

Subset posterior aggregation. To show the effectiveness of our algorithm in real-world applications, we apply our method to aggregate subset posterior distributions using barycenters, which has been shown as an effective alternative to the full posterior in the massive data setting [Sri+15; SLD18; Sta+17]. We consider Poisson regression for the task of predicting the hourly number of bike rentals using features such as the day of the week and weather conditions.[3] We use one intercept and 8 regression coefficients for the Poisson model, and consider the posterior on the 8-dimensional regression coefficients. We randomly split the data into 5 equally-sized subsets and obtain 10^5 samples from each subset posterior using the Stan library [Car+17].

The barycenter of subset posteriors converges to the full data posterior [SLD18]. Hence, to evaluate the quality of the barycenter computed from the subset posterior samples, we use the full posterior samples as the ground truth and report the differences in covariance using sufficiently many samples from the barycenter, and compare against other free-support barycenter algorithms (Tab. 2). See Sec. 4.A.2 for more details. To show how the quality of the barycenter improves as more samples are used from our

[3] http://archive.ics.uci.edu/ml/datasets/Bike+Sharing+Dataset

Table 2: Comparison of subset posterior aggregation results in the covariance difference $\|\Sigma - \Sigma^*\|$, where Σ is the covariance of the barycenter samples from each method, and Σ^* is that of the full posterior. All experiments are repeated 20 times with the mean and standard deviation reported. As in Tab. 1, we use 5000 support points in [CD14] and 100 support points in [CCS18] as these are the maximum numbers permitted for the algorithms to terminate in a reasonable amount of time.

[CD14]	[CCS18]	Ours with (d) and R_{L_2}
$2.56\times10^{-7}(2.17\times10^{-9})$	$9.37\times10^{-4}(4.84\times10^{-5})$	$\mathbf{2.43\times10^{-7}(6.57\times10^{-8})}$

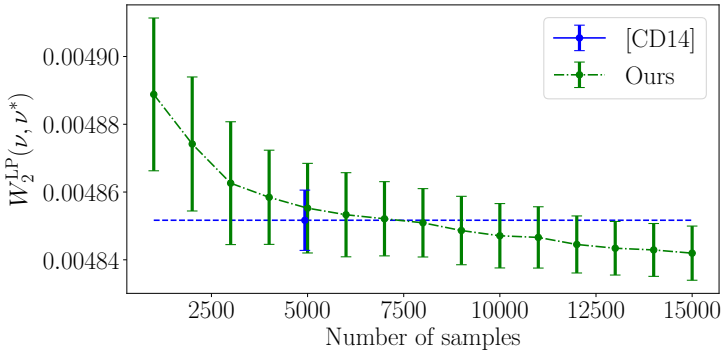

Figure 3: 2-Wasserstein distance versus the number of samples from the output of our algorithm with (d). Same number of points are used from both the full posterior and the computed barycenter to compute $W_2^{\mathrm{LP}}(\nu, \nu^*)$. The blue bar is the result of [CD14] with 5000 support points. The caps around each solid dot indicate the standard deviation across 20 independent trials.

barycenter, we plot the 2-Wasserstein distance versus the number of samples in Fig. 3. Since computing $W_2(\nu, \nu^*)$ requires solving a large linear program, we are only able to produce the result up to 15000 samples. This is also a limitation in [CD14] as each iteration of their alternating optimization solves many large linear programs; for this reason we are only able to use 5000 support points for their method. We see that as we use more samples, W_2 steadily decreases with lower variance, and we expect the decrease to continue with more samples. With 15000 samples our barycenter is closer to the full posterior than that of [CD14].

■ 4.6 Conclusion and Future Directions

Our stochastic algorithm computes the barycenter of continuous distributions without discretizing the output barycenter, and has been shown to provide a clear advantage

over past methods in higher dimensions. However the performance of our algorithm still suffers from a curse of dimensionality when the dimension is too high. Indeed in this case the support measure we fix from the beginning becomes a poor proxy for the true barycenter, and an enormous batch size is required to evaluate the expectation in (4.16) with reasonably small variance. One future direction is to find a way to estimate the support measure dynamically, but choosing a representation for this task is challenging. Another issue that can be addressed is to reduce regularization bias. This can either happen by formulating alternative versions of the dual problem or by improving the methods for estimating a Monge map.

Appendices

■ 4.A Experimental Details and Additional Results

♢ 4.A.1 Multivariate Gaussians with varying dimensions

We generate the multivariate Gaussians in dimension d used in Table 1 in the following manner. The mean is chosen uniformly at random in $[-1, 1]^d$. The covariance matrix is obtained by first sampling a matrix A with uniform entries in $[-0.3, 0.3]$ and then taking $AA^\top$ as the covariance matrix. We reject A if its condition number (computed with respect to 2-norm) is not in $[2, 80]$.

We show in Table 4.A.1 additional results for our algorithm with different choices of Monge map estimation methods and regularizers; in the last column we show the result of [CD14] where we use Sinkhorn algorithm [Cut13] instead of LP (see Table 1 for results with LP) to obtain the transport plan at every iteration.

Table 4.A.1: Additional results for the multivariate Gaussian experiment. Reported are the covariance difference $\|\Sigma - \Sigma^*\|_F$ where Σ is the MLE covariance of the barycenter computed by each method, Σ^* is the ground truth covariance, and $\|\cdot\|_F$ is the Frobenius norm. Smaller is better. All experiments are repeated 5 times with the mean and standard deviation reported. Here R_{L_2} refers to quadratic regularization with $\epsilon = 10^{-4}$, and R_e refers to entropic regularization with $\epsilon = 0.1$. The regularizing ϵ is further scaled with respect to the diagonal length of the bounding box squared. For [CD14] with Sinkhorn algorithm, we choose $\epsilon = 0.1$.

d	Ours with (d) and R_{L_2}	Ours with (e) and R_{L_2}	Ours with (d) and R_e	[CD14] with Sinkhorn
2	$\mathbf{1.98 \times 10^{-3} (1.17 \times 10^{-4})}$	$2.38 \times 10^{-3} (2.48 \times 10^{-4})$	$8.25 \times 10^{-3} (5.02 \times 10^{-4})$	$5.22 \times 10^{-2} (5.09 \times 10^{-4})$
3	$\mathbf{5.05 \times 10^{-3} (6.32 \times 10^{-4})}$	$5.70 \times 10^{-3} (6.90 \times 10^{-4})$	$8.15 \times 10^{-3} (6.50 \times 10^{-4})$	$7.46 \times 10^{-2} (3.87 \times 10^{-4})$
4	$\mathbf{1.22 \times 10^{-2} (1.44 \times 10^{-3})}$	$1.27 \times 10^{-2} (1.19 \times 10^{-3})$	$2.06 \times 10^{-2} (7.40 \times 10^{-4})$	$8.78 \times 10^{-2} (1.40 \times 10^{-3})$
5	$\mathbf{1.52 \times 10^{-2} (1.18 \times 10^{-3})}$	$2.33 \times 10^{-2} (2.86 \times 10^{-3})$	$3.72 \times 10^{-2} (9.81 \times 10^{-4})$	$1.00 \times 10^{-1} (7.30 \times 10^{-4})$
6	$\mathbf{2.37 \times 10^{-2} (3.24 \times 10^{-3})}$	$3.27 \times 10^{-2} (2.63 \times 10^{-3})$	$6.13 \times 10^{-2} (2.69 \times 10^{-3})$	$1.10 \times 10^{-1} (7.93 \times 10^{-4})$
7	$\mathbf{4.07 \times 10^{-2} (2.65 \times 10^{-3})}$	$4.83 \times 10^{-2} (2.90 \times 10^{-3})$	$8.42 \times 10^{-2} (4.62 \times 10^{-4})$	$1.16 \times 10^{-1} (5.44 \times 10^{-4})$
8	$\mathbf{4.23 \times 10^{-2} (3.14 \times 10^{-3})}$	$4.79 \times 10^{-2} (2.46 \times 10^{-3})$	$1.20 \times 10^{-1} (2.38 \times 10^{-3})$	$1.18 \times 10^{-1} (7.07 \times 10^{-4})$

To briefly comment on the runtime of our algorithm (with (d)) and that of [CD14] and [CCS18], in the 8-dimensional Gaussian experiment from Table 1, our algorithm takes around 15 minutes, [CD14] takes 20 minutes, while [CCS18] takes an hour or longer. The simple form of Algorithm 4.4.1 and the convex nature of (4.16) give rise to fast convergence of our approach.

♢ 4.A.2 Subset posterior aggregation

We adopted the `BikeTrips` dataset and preprocessing steps from https://github.com/trevorcampbell/bayesian-coresets [CB19]. The posterior samples in the subset posterior aggregation experiment are generated using NUTS sampler [HG+14] implemented by the Stan library [Car+17]. To enforce appropriate scaling of the prior in the subset posteriors we use stochastic approximation trick [SLD18], i.e. scaling the log-likelihood by the number of subsets. Please see the code for further details.

In Table 4.A.2, we show additional results comparing [CD14] and our algorithm in three different losses: difference in mean, covariance, and the (unregularized) 2-Wasserstein distance computed using 5000 samples. See Figure 3 for a comparison with varying number of samples used to compute the 2-Wasserstein distance.

Table 4.A.2: Comparison of subset posterior aggregation results in difference in mean, covariance, and 2-Wasserstein distance. All experiments are repeated 20 times with the mean and standard deviation reported. Variables with a superscript star (μ^*, Σ^*, ν^*) are quantities from the full posterior, and variables without a star are from the computed barycenter. The mean and covariance are estimated with sufficiently many samples from the barycenter, while the 2-Wasserstein distance is computed using 5000 samples from both the barycenter and the full posterior.

Loss	[CD14]	Ours with (d) and R_{L_2}	Ours with (e) and R_{L_2}
$\|\mu - \mu^*\|$	$4.79\times10^{-3}(3.19\times10^{-6})$	$4.79\times10^{-3}(5.96\times10^{-7})$	$\mathbf{4.79\times10^{-3}(1.80\times10^{-7})}$
$\|\Sigma - \Sigma^*\|$	$2.56\times10^{-7}(2.17\times10^{-9})$	$\mathbf{2.43\times10^{-7}(6.57\times10^{-8})}$	$9.51\times10^{-7}(6.62\times10^{-9})$
$W_2^{\mathrm{LP}}(\nu, \nu^*)$	$\mathbf{4.85\times10^{-3}(8.90\times10^{-6})}$	$4.86\times10^{-3}(9.40\times10^{-6})$	$4.96\times10^{-3}(6.10\times10^{-6})$

Chapter 5

Learning Proximal Operators to Discover Multiple Optima

In Chapter 4, we saw how convex duality can be used to derive feasible formulation with neural network parameterization. In this chapter, we show how non-convex classical optimization with multiple global minima can be solved using a *convex* functional formulation where minima are encoded as a pushforward generative model. In addition, we show global convergence of training the neural network under overparameterization. This chapter is based on the publication [Li+23a].

5.1 Introduction

Searching for multiple optima of an optimization problem is a ubiquitous yet underexplored task. In applications like low-rank recovery [GJZ17], topology optimization [PFS21], object detection [Lin+14], and symmetry detection [Shi+20], it is desirable to recover multiple near-optimal solutions, either because there are many equally-performant global optima or due to the fact that the optimization objective does not capture user preferences precisely. Even for single-solution non-convex optimization, typical methods look for multiple local optima from random initial guesses before picking the best local optimum. Additionally, it is often desirable to obtain solutions to a family of optimization problems with parameters not known in advance, for instance, the weight of a regularization term, without having to restart from scratch.

Formally, we define a *multi-solution optimization* (MSO) problem to be the minimization $\min_{x\in\mathcal{X}} f_\tau(x)$, where $\tau \in \mathcal{T}$ encodes parameters of the problem, $\mathcal{X}$ is the search space of the variable x, and $f_\tau : \mathbb{R}^d \to \mathbb{R}$ is the objective function depending on τ. The goal of MSO is to identify multiple solutions for each $\tau \in \mathcal{T}$, i.e., the set $\{x^* \in \mathcal{X} : f_\tau(x^*) = \min_{x\in\mathcal{X}} f_\tau(x)\}$, which can contain more than one element or even infinitely many elements. In this work, we assume that $\mathcal{X} \subset \mathbb{R}^d$ is bounded and that

d is small, and that $\mathcal{T}$ is, in a loose sense, a continuous space, such that the objective f_τ changes continuously as τ varies. To make gradient-based methods viable, we further assume that each f_τ is differentiable almost everywhere. As finding all global minima in the general case is extremely challenging, realistically our goal is to find a diverse set of local minima.

As a concrete example, for object detection, $\mathcal{T}$ could parameterize the space of images and $\mathcal{X}$ could be the 4-dimensional space of bounding boxes (ignoring class labels). Then, $f_\tau(x)$ could be the minimum distance between the bounding box $x \in \mathcal{X}$ and any ground truth box for image $\tau \in \mathcal{T}$. Minimizing $f_\tau(x)$ would yield *all* object bounding boxes for image τ. Object detection can then be cast as solving this MSO on a training set of images and extrapolating to unseen images (Sec. 5.5.5). Object detection is a singular example of MSO where the ground truth annotation is widely available. In such cases, supervised learning can solve MSO by predicting a fixed number of solutions together with confidence scores using a set-based loss such as the Hausdorff distance. Unfortunately, such annotation is not available for most optimization problems in the wild where we only have access to the objective functions—this is the setting that our method aims to tackle.

Our work is inspired by the **proximal-point algorithm** (PPA), which applies the **proximal operator** of the objective function to an initial point iteratively to refine it to a local minimum. PPA is known to converge faster than gradient descent even when the proximal operator is approximated, both theoretically [Roc76; Roc21] and empirically (e.g., Figure 2 of Hoheisel, Laborde, and Oberman [HLO20]). If the proximal operator of the objective function is available, then MSO can be solved efficiently by running PPA from a variety of initial points. However, obtaining a good approximation of the proximal operator for generic functions is difficult, and typically we have to solve a separate optimization problem for each evaluation of the proximal operator [DG19].

In this work, we approximate the proximal operator using a neural network that is trained using a straightforward loss term including only the objective and a *proximal term* that penalizes deviation from the input point. Crucially, our training does not require accessing the ground truth proximal operator. Additionally, neural parameterization allows us to learn the proximal operator for all $\{f_\tau\}_{\tau \in \mathcal{T}}$ by treating τ as an input to the network along with an application-specific encoder. Once trained, the learned proximal operator allows us to effortlessly run PPA from any initial point to arrive at a nearby local minimum; from a generative modeling point of view, the learned proximal operator implicitly encodes the solutions of an MSO problem as the pushforward of a prior distribution by iterated application of the operator. Such a formulation bypasses the need to predict a fixed number of solutions and can represent infinitely many solutions. The proximal term in our loss promotes the convexity of the formulation: applying recent results [KH19], we show that for weakly-convex objectives with Lipschitz gradients—in particular, objectives with bounded second derivatives—with practical degrees of overparameterization, training converges globally and the ground truth proximal operator is

recovered (Thm. 5.3.1 below). Such a global convergence result is not known for any previous learning-to-optimize method [Che+22a].

Literature on MSO is scarce, so we build a benchmark with a wide variety of applications including level set sampling, non-convex sparse recovery, max-cut, 3D symmetry detection, and object detection in images. When evaluated on this benchmark, our learned proximal operator reliably produces high-quality results compared to reasonable alternatives, while converging in a few iterations.

■ 5.2 Related Works

Learning to optimize. Learning-to-optimize (L2O) methods utlilize past optimization experience to optimize future problems more effectively; see [Che+22a] for a survey. *Model-free* L2O uses recurrent neural networks to discover new optimizers suitable for similar problems [And+16; LM16; Che+17; Cao+19]; while shown to be practical, these methods have almost no theoretical guarantee for the training to converge [Che+22a]. In comparison, we learn a problem-dependent proximal operator so that at test time we do not need access to objective functions or their gradients, which can be costly to evaluate (e.g. symmetry detection in Sec. 5.5.4) or unavailable (e.g. object detection in Sec. 5.5.5). *Model-based* L2O substitutes components of a specialized optimization framework or schematically unrolls an optimization procedure with neural networks. Related to proximal methods, Gregor and LeCun [GL10] emulate a few iterations of proximal gradient descent using neural networks for sparse recovery with an ℓ^1 regularizer, extended to non-convex regularizers by Yang et al. [Yan+20]; a similar technique is applied to susceptibility-tensor imaging in Fang et al. [Fan+23]. Gilton, Ongie, and Willett [GOW21] propose a deep equilibrium model with proximal gradient descent for inverse problems in imaging that circumvents expensive backpropagation of unrolling iterations. Meinhardt et al. [Mei+17] use a fixed denoising neural network as a surrogate proximal operator for inverse imaging problems. All these works use schematics of proximal methods to design a neural network that is then trained with strong supervision. In contrast, we learn the proximal operator directly, requiring only access to the objectives; we do not need ground truth for inverse problems during training.

Existing L2O methods are not designed to recover multiple solutions: without a proximal term like in (5.2), the learned operator can degenerate even with multiple starts (Sec. 5.D.3).

Finding multiple solutions. Many heuristic methods have been proposed to discover multiple solutions including niching [BEB07; Li09], parallel multi-starts [LW18], and deflation [PFS21]. However, all these methods do not generalize to similar but unseen problems.

Predicting multiple solutions at test time is universal in deep learning tasks like multi-label classification [TK07] and detection [Liu+20]. The typical solution is to ask the network to predict a fixed number of candidates along with confidence scores to indicate how likely each candidate is a solution [Ren+15; Li+19; Car+20]. Then the solutions will be chosen from the candidates using heuristics such as non-maximum suppression [NV06]. Models that output a fixed number of solutions without taking into account the unordered set structure can suffer from "discontinuity" issues: a small change in set space requires a large change in the neural network outputs [ZHP19]. Furthermore, this approach cannot handle the case when the solution set is continuous.

Wasserstein gradient flow. Our formulation (5.2) corresponds to one step of JKO discretization of the Wasserstein gradient flow where the energy functional is the the linear functional dual to the MSO objective function [JKO98; Ben+16]. See the details in Sec. 5.E. Compared to recent works on neural Wasserstein gradient flows [Mok+21; Par+23; Bun+22], where a separate network parameterizes the pushforward map for every JKO step, our functional's linearity makes the pushforward map identical for each step, allowing end-to-end training using a single neural network. We additionally let the network input a parameter τ, in effect learning a continuous family of JKO-discretized gradient flows.

■ 5.3 Method

◇ 5.3.1 Preliminaries

Given the objective $f_\tau : \mathbb{R}^d \to \mathbb{R}$ of an MSO problem parameterized by τ, the corresponding **proximal operator** [Mor62; Roc76; PB14] is defined, for a fixed $\lambda \in \mathbb{R}_{>0}$, as

$$\operatorname{prox}(x;\tau) \triangleq \arg\min_y \left\{ f_\tau(y) + \frac{\lambda}{2}\|y-x\|_2^2 \right\}. \tag{5.1}$$

The weight λ in the **proximal term** $\frac{\lambda}{2}\|y-x\|_2^2$[1] controls how close $\operatorname{prox}(x;\tau)$ is to x: increasing λ will reduce $\|\operatorname{prox}(x;\tau)-x\|_2$. For the arg min in (5.1) to be unique, a sufficient condition is that f_τ is ξ-weakly convex with $\xi < \lambda$, so that $f_\tau(y) + \frac{\lambda}{2}\|y-x\|^2$ is strongly convex. The class of weakly convex functions is deceivingly broad: for instance, any twice differentiable function with bounded second derivatives (e.g. any C^2 function on a compact set) is weakly convex. When the function is convex, $\operatorname{prox}(x;\tau)$ is precisely one

[1]The usual convention is to use the reciprocal of λ in front of the proximal term. We use a different convention to associate λ with the convexity of (5.1).

step of the backward Euler discretization of integrating the vector field $-\nabla f_\tau$ with time step $\frac{1}{\lambda}$ (see Section 4.1.1 of Parikh and Boyd [PB14]).

The **proximal-point algorithm** (PPA) for finding a local minimum of f_τ iterates

$$x^k \triangleq \text{prox}(x^{k-1}; \tau), \forall k \in \mathbb{N}_{\geq 1},$$

with initial point x^0 [Roc76]. In practice, $\text{prox}(x; \tau)$ often can only be approximated, resulting in *inexact* PPA. When the objective function is locally indistinguishable from a convex function and x^0 is sufficiently close to the set of local minima, then with reasonable stopping criterion, inexact PPA converges linearly to a local minimum of the objective: the smaller λ is, the faster the convergence rate becomes (Theorem 2.1-2.3 of Rockafellar [Roc21]).

♢ 5.3.2 Learning proximal operators

The fast convergence rate of PPA makes it a strong candidate for MSO: to obtain a diverse set of solutions for any $\tau \in \mathcal{T}$, we only need to run a few iterations of PPA from random initial points. The proximal term penalizes big jumps and prevents points from collapsing to a single solution. However, running a subroutine to approximate $\text{prox}(x; \tau)$ for every pair (x, τ) can be costly.

To overcome this issue, we *learn* the operator $\text{prox}(\cdot; \cdot)$ given access to $\{f_\tau\}_{\tau \in \mathcal{T}}$. A naïve way to learn $\text{prox}(\cdot; \cdot)$ is to first solve (5.1) to produce ground truth for a large number of (x, τ) pairs independently using gradient-based methods and then learn the operator using mean-squared error loss. However, this approach is costly as the space $\mathcal{X} \times \mathcal{T}$ can be large. Moreover, this procedure requires a stopping criterion for the minimization in (5.1), which is hard to design *a priori*.

Instead, we formulate the following end-to-end optimization over the space of functions:

$$\min_{\Phi: \mathcal{X} \times \mathcal{T} \to \mathcal{X}} \mathbb{E}_{\substack{x \sim \mu \\ \tau \sim \nu}} \left[f_\tau(\Phi(x, \tau)) + \frac{\lambda}{2} \|\Phi(x, \tau) - x\|_2^2 \right], \tag{5.2}$$

where x is sampled from μ, a distribution on $\mathcal{X}$, and τ is sampled from ν, a distribution on $\mathcal{T}$. To get (5.2) from (5.1), we essentially substitute y with the output $\Phi(x, \tau)$ and integrate over the product probability distribution $\mu \otimes \nu$.

To solve (5.2), we parameterize $\Phi : \mathcal{X} \times \mathcal{T} \to \mathcal{X}$ using a neural network with additive and multiplicative residual connections (Sec. 5.B). Intuitively, the implicit regularization of neural networks aligns well with the regularity of $\text{prox}(\cdot; \cdot)$: for a fixed τ the proximal operator $\text{prox}(\cdot; \tau)$ is 1-Lipschitz in local regions where f_τ is convex, while as the parameter τ varies f_τ changes continuously so $\text{prox}(x; \tau)$ should not change too much. To make (5.2) computationally practical during training, we realize ν as a training dataset. For

the choice of μ, we employ an importance sampling technique from Wang and Solomon [WS19] as opposed to using Uniform($\mathcal{X}$), the uniform distribution over $\mathcal{X}$, so that the learned operator can refine near-optimal points (Sec. 5.C). To train Φ, we sample a mini-batch of (x, τ) to evaluate the expectation and optimize using Adam [KB14]. For problems where the space $\mathcal{T}$ is structured (e.g. images or point clouds), we first embed τ into a Euclidean feature space through an encoder before passing it to Φ. Such encoder is trained together with operator network Φ. This allows us to use efficient domain-specific encoder (e.g. convolutional networks) to facilitate generalization to unseen τ.

To extract multiple solutions at test time for a problem with parameter τ, we sample a batch of x's from Uniform($\mathcal{X}$) and apply the learned $\Phi(\cdot, \tau)$ to the batch of samples a few times. Each application of Φ approximates a single step of PPA. From a distributional perspective, for $k \in \mathbb{N}_{\geq 0}$, we can view Φ^k—the operator Φ applied k times—as a generative model so that the pushforward distribution, $(\Phi^k)_{\#}(\text{Uniform}(\mathcal{X}))$, concentrates on the set of local minima approximates as k increases. An advantage of our representation is that it can represent arbitrary number of solutions even when the set of minima is continuous (Fig. 2). This procedure differs from those in existing L2O methods [Che+22a]: at test time, we do not need access to $\{f_\tau\}_{\tau \in \mathcal{T}}$ or their gradients, which can be costly to evaluate or unavailable; instead we only need τ (e.g. in the case of object detection, τ is an image).

♢ 5.3.3 Convergence of training

We have turned the problem of finding multiple solutions for each f_τ in the space $\mathcal{X}$ into the problem of finding a single solution for (5.2) in the space of functions. If the f_τ's are ξ-weakly convex with $\xi < \lambda$ and μ, ν have full support, then the arg min in (5.1) is unique for every pair (x, τ) and hence the functional solution of (5.2) is the unique proxmal operator $\text{prox}(\cdot; \tau)$.

If in addition the gradients of the objectives are Lipschitz, using recent learning theory results [KH19] we can show that with practical degrees of over-parameterization, gradient descent on neural network parameters of Φ converges globally during training. Suppose our training dataset is $S = \{(x_i, \tau_i)\}_{i=1}^n \subset \mathcal{X} \times \mathcal{T}$. Define the training loss, a discretized version of (5.2) using S, to be, for $g : \mathcal{X} \times \mathcal{T} \to \mathcal{X}$,

$$L(g) \triangleq \frac{1}{n} \sum_{i=1}^{n} \left[f_{\tau_i}(g(x_i, \tau_i)) + \frac{\lambda}{2} \|g(x_i, \tau_i) - x_i\|_2^2 \right]. \tag{5.3}$$

Theorem 5.3.1 (informal)**.** *Suppose for any $\tau \in \mathcal{T}$, the objective f_τ is differentiable, ξ-weakly convex, and ∇f_τ is ζ-Lipschitz with $\xi \leq \lambda$. Then there exists a feed-forward neural network with $\tilde{\Omega}(n)$ total parameters[2] and common activation units, such that, when the initial weights are drawn from a Gaussian distribution, with high probability, gradient*

[2]We use $\tilde{\Omega}$ notation in the standard way, i.e., $f \in \tilde{\Omega}(n) \iff \exists k \in \mathbb{N}_{\geq 0}$ such that $f \in \Omega(n \log^k n)$.

descent on its weights using a fixed learning rate will eventually reach the minimum loss $\min_{g:\mathcal{X}\times\mathcal{T}\to\mathcal{X}} L(g)$. *The number of iterations needed to achieve* $\epsilon > 0$ *training error is* $O((\lambda+\zeta)/\epsilon)$, *and when this occurs, if* $\xi < \lambda$, *then the mean-squared error of the learned proximal operator compared to the true one is* $O(\frac{2\epsilon}{(\lambda-\xi)})$ *on training data.*

We state and prove Thm. 5.3.1 formally in Sec. 5.A. Even though the optimization over network weights is non-convex, training can still result in a globally minimal loss and the true proximal operator can be recovered. In Sec. 5.D.2, we empirically verify that when the objective is the ℓ^1 norm, the trained operator converges to the true proximal operator, the shrinkage operator. In Sec. 5.D.3, we study the effect of λ in relation to the weakly-convex constant ξ for the 2D cosine problem and compare to an L2O particle-swarm method [Cao+19].

We note a few gaps between Thm. 5.3.1 and our implementation. First, we use SGD with mini-batching instead of gradient descent. Second, instead of feed-forward networks, we use a network architecture with residual connections (Fig. 5.B.1), which works better empirically. Under these conditions, global convergence results can still be obtained, e.g., via [ALS19, Theorems 6 and 8], but with large polynomial bounds in n, H for the network parameters. Another gap is caused by the restriction of the function class of the objectives. In several applications in Sec. 5.5, the objective functions are not weakly convex or have Lipschitz gradients, or we deliberately choose small λ for faster PPA convergence; we empirically demonstrate that our method remains effective.

■ 5.4 Performance Measures

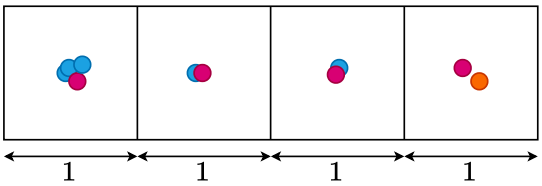

Figure 1: Interpretation of D_t. In this example, the witness W is drawn uniformly from the union of four squares. If A_t (resp. B_t) is the set of red (resp. blue) points, then $\Pr(D_t \approx 0) = \frac{3}{4}$ and $\Pr(D_t \approx 0.5) = \frac{1}{4}$, since D_t is only non-zero when W is in the rightmost square. This aligns well with the intuition that $\frac{3}{4}$ of the red points match with the blue ones. In comparison, the Hausdorff distance between A_t and B_t is approximately 1, which is the same as the Hausdorff distance between the orange point and B_t, despite the fact most of red points are close to the blue ones.

Metrics. Designing a single-valued metric for MSO is challenging since one needs to consider the diversity of the solutions as well each solution's level of optimality. For an MSO problem with parameter τ and objective f_τ, the output of an MSO algorithm can be represented as a (possibly infinite) set of solutions $\{x_\alpha\}_\alpha \subset \mathcal{X}$ with objective values $u_\alpha \triangleq f_\tau(x_\alpha)$. Suppose we have access to ground truth solutions $\{y_\beta\}_\beta \subset \mathcal{X}$ with $v_\beta \triangleq f_\tau(y_\beta)$. Pick a threshold $t \in \mathbb{R}$ and denote $A_t \triangleq \{x_\alpha : u_\alpha \leq t\}, B_t \triangleq \{y_\beta : v_\beta \leq t\}$. Let W be a random variable that is uniformly distributed on $\mathcal{X}$. Define a random variable

$$D_t \triangleq \frac{1}{2}\|\pi_{A_t}(W) - \pi_{B_t}(\pi_{A_t}(W))\|_2 + \frac{1}{2}\|\pi_{B_t}(W) - \pi_{A_t}(\pi_{B_t}(W))\|_2,$$

where $\pi_S(x) \triangleq \arg\min_{s\in S}\|x-s\|_2$. We call W a *witness* of D_t, as it witnesses how different A_t and B_t are near W. To summarize the law of D_t, we define the *witnessed divergence* and *witnessed precision at* $\delta > 0$ as

$$\mathrm{WD}_t \triangleq \mathbb{E}[D_t] \quad \text{and} \quad \mathrm{WP}_t^\delta \triangleq \Pr(D_t < \delta). \tag{5.4}$$

Witnesses help handle unbalanced clusters that can appear in the solution sets. These metrics are agnostic to duplicates, unlike the chamfer distance or optimal transport metrics. Compared to alternatives like the Hausdorff distance, WD_t remains low if a small portion of A_t, B_t are mismatched. We illustrate these metrics in Fig. 1. One can interpret WD_t as a weighted chamfer distance whose weight is proportional to the volume of the ℓ^2-Voronoi cell at each point in either set.

Particle Descent: Ground Truth Generation. A naïve method for MSO is to run gradient descent until convergence on randomly sampled particles in $\mathcal{X}$ for every $\tau \in \mathcal{T}$. We use this method to generate approximated ground truth solutions to compute the metrics in (5.4) when the ground truth is not available. This method is not directly comparable to ours since it cannot generalize to unseen τ's at test time. Remarkably, for highly non-convex objectives, particle descent can produce worse solutions than the ones obtained using the learned proximal operator (Fig. 5.D.7).

Learning Gradient Descent Operators. As there is no readily-available application-agnostic baseline for MSO, we propose the following method that learns iterations of the gradient descent operator. Fix $Q \in \mathbb{N}_{\geq 1}$ and a step size $\eta > 0$. We optimize an operator Ψ via

$$\min_{\Psi:\mathcal{X}\times\mathcal{T}\to\mathcal{X}} \mathbb{E}_{\substack{x\sim\mu\\ \tau\sim\nu}}\|\Psi(x,\tau) - \Psi_Q^*(x;\tau)\|_2^2, \tag{5.5}$$

where $\Psi_Q^*(x;\tau)$ is the result of Q steps of gradient descent on f_τ starting at x, i.e., $\Psi_0^*(x;\tau) = x$, and $\Psi_k^*(x;\tau) = \Psi_{k-1}^*(x;\tau) - \eta\nabla f_\tau(\Psi_{k-1}^*(x;\tau))$. Each iteration of minimizing (5.5) requires Q evaluations of ∇f_τ, which can be costly (e.g., for symmetry detection

in Sec. 5.5.4). We use importance sampling similar to Sec. 5.C. An ODE interpretation is that Ψ performs Q iterations of *forward* Euler on the gradient field ∇f_τ, whereas the learned proximal operator performs a single iteration of *backward* Euler. We choose $Q = 10$ for all experiments except for symmetry detection (Sec. 5.5.4) where we choose $Q = 1$ because otherwise the training will take > 200 hours. As we will see in Fig. 5.D.6, aside from slower training, this approach struggles with non-smooth objectives due to the fixed step size η, while the learned proximal operator has no such issues.

■ 5.5 Applications

We consider five applications to benchmark our MSO method, chosen to highlight the ubiquity of MSO in diverse settings. We abbreviate POL for proximal operator learning (proposed method), GOL for gradient operator learning (Sec. 5.4), and PD for particle descent (Sec. 5.4). Further details about each application can be found in Sec. 5.D. The source code for all experiments can be found at https://github.com/lingxiaoli94/POL.

♢ 5.5.1 Sampling from level sets

Formulation. Level sets provide a concise and resolution-free implicit shape representation [Mus+02; Par+19; Sit+20]. Yet they are less intuitive to work with, even for straightforward tasks on discretized domains (meshes, point clouds) like visualizing or integration on the domain. We present an MSO formulation to sample from level sets, enabling the adaptation of downstream tasks to level sets.

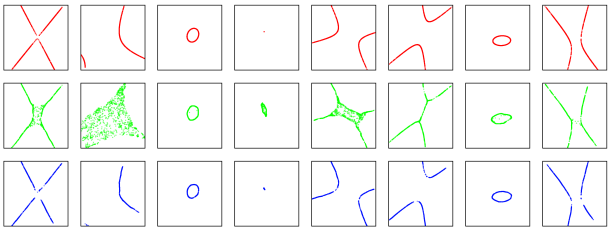

Figure 2: Visualization of the solutions for the conic section problem. Red, green, and blue indicate the solutions by PD, GOL, and POL respectively. See Fig. 5.D.3 for more examples.

Given a family of functions $\{g_\tau : \mathcal{X} \to \mathbb{R}^q\}_{\tau \in \mathcal{T}}$, for each τ suppose we want to sample from the 0-level set $g_\tau^{-1}(0)$. We formulate an MSO problem with objective $f_\tau(x) \triangleq \|g_\tau(x)\|_2^2$, whose global optima are precisely $g_\tau^{-1}(0)$. We do not need assumptions on level set topology or that the implicit function represents a distance field, unlike most existing methods [Par+19; Den+20; CTZ20].

Benchmark. We consider sampling from conic sections. We keep this experiment simple so as to visualize the solutions easily. Let $\mathcal{X} = [-5, 5]^2$ and $\mathcal{T} = [-1, 1]^6$. For $\tau = (A, B, C, D, E, F) \in \mathcal{T}$, define g_τ to be $g_\tau(x_1, x_2) \triangleq Ax^2 + Bxy + Cy^2 + Dx + Ey + F$. Since $f_\tau = (g_\tau)^2$ is a defined on a compact $\mathcal{X}$, it satisfies the conditions of Thm. 5.3.1 for

a large λ, but a large λ corresponds to small PPA step size. Empirically, small λ for POL gave decent results compared to GOL: Fig. 2 illustrates that POL consistently produces sharper level sets for both hyperbolas ($B^2 - 4AC > 0$) and ellipses ($B^2 - 4AC < 0$). Fig. 5.D.4 shows that POL yields significantly higher WP_t^δ than GOL for small δ, implying that details are well recovered. Fig. 5.D.5 verifies that iterating the trained operator of POL converges much faster than that of GOL. It is straightforward to extend this setting to sample from more complicated implicit shapes parameterized by τ.

◇ 5.5.2 Sparse recovery

Formulation. In signal processing, the *sparse recovery* problem aims to recover a signal $x^* \in \mathcal{X} \subset \mathbb{R}^d$ from a noisy measurement $y \in \mathbb{R}^m$ distributed according to $y = Ax^* + e$, where $A \in \mathbb{R}^{m \times d}$, $m < d$, and e is measurement noise [BT09]. In applications like imaging and speech recognition, the signals are *sparse*, with few non-zero entries [Mar+18]. Hence, the goal of sparse recovery is to recover a sparse x^* given A and y.

A common way to encourage sparsity is to solve least-squares plus an ℓ^p norm on the signal:

$$\min_{x \in \mathcal{X}} \|Ax - y\|_2^2 + \alpha \|x\|_p^p, \tag{5.6}$$

for $\alpha, p > 0$ and $\|x\|_p^p \triangleq \sum_{i=1}^d (x_i^2 + \epsilon)^{p/2}$ for a small ϵ to prevent instability. We consider the non-convex case where $0 < p < 1$. Compared to convex alternatives like in LASSO ($p = 1$), non-convex ℓ^p norms require milder conditions under which the global optima of (5.6) are the desired sparse x^* [CS08; CG14].

To apply our MSO framework, we define $\tau = (\alpha, p) \in \mathcal{T}$ and f_τ to be the objective (5.6) with corresponding α, p. Compared to existing methods for non-convex sparse recovery [LXY13], our method can recover multiple solutions from the non-convex landscape for a family of α's and p's without having to restart. The user can adjust parameters α, p to quickly generate candidate solutions before choosing a solution based on their preference.

Benchmark. Let $\mathcal{X} = [-2, 2]^8, \mathcal{T} = [0, 1] \times [0.2, 0.5]$. We consider highly non-convex ℓ^p norms with $p \in [0.2, 0.5]$ to test our method's limits. We choose $d = 8$ and $m = 4$, and sample the sparse signal x^* uniformly in $\mathcal{X}$ with half of the coordinates set to 0. We then sample entries in A i.i.d. from $\mathcal{N}(0, 1)$ and generate $y = Ax^* + e$ where $e \sim \mathcal{N}(0, 0.1)$. Although $\|x\|_p^p$ is not weakly convex, POL achieves decent results (Fig. 5.D.6). Notably, POL often reaches a better objective than PD (Fig. 5.D.7) while retaining diversity, even though POL uses a much bigger step size ($\frac{1}{\lambda} = 0.1$ compared to PD's 10^{-5}) and needs to learn a different operator for an entire family of $\tau \in \mathcal{T}$. In Fig. 5.D.8, we additionally compare POL with proximal gradient descent [Tib+10] for $p = \frac{1}{2}$ where the corresponding thresholding formula has a closed-form [CSX13]. Remarkably, we have observed superior performance of POL against such a strong baseline.

♢ 5.5.3 Rank-2 relaxation of max-cut

Formulation. MSO can be applied to solve combinatorial problems that admit smooth non-convex relaxations. Here, we consider the classical problem of finding the maximum cut of an undirected graph $G = (V, E)$, where $V = \{1, \dots, n\}$, $E \subset V \times V$, with edge weights $\{w_{ij}\} \subset \mathbb{R}$ so that $w_{ij} = 0$ if $(i, j) \notin E$. The goal is to find $\{x_i\} \in \{-1, +1\}^V$ to maximize $\sum_{i,j} w_{ij}(1 - x_i x_j)$.

Burer, Monteiro, and Zhang [BMZ02] propose solving $\min_{\theta \in \mathbb{R}^n} \sum_{i,j} w_{ij} \cos(\theta_i - \theta_j)$, a rank-2 non-convex relaxation of the max-cut problem. This objective inherits weak convexity from cosine, so it satisfies the conditions of Thm. 5.3.1. In practice, instead of using angles as the variables which are ambiguous up to 2π, we represent each variable as a point on the unit circle S^1, so we choose $\mathcal{X} = (S^1)^n$ and $\mathcal{T}$ be the space of all edge weights with n vertices. For $\tau = \{\tau_{ij}\} \in \mathcal{T}$ corresponding to a graph with edge weights $\{\tau_{ij}\}$, we define, for $x \in \mathcal{X}$,

$$f_\tau(x) \triangleq \sum_{i,j} \tau_{ij} x_i^\top x_j. \tag{5.7}$$

After minimizing f_τ, we can find cuts using a Goemans and Williamson-type procedure (1995). Instead of using heuristics to find optima near a solution [BMZ02], our method can help the user effortlessly explore the set of near-optimal solutions without hand-designed heuristics.

Benchmark. We apply our formulation to K_8, the complete graph with 8 vertices. Hence $\mathcal{X} = (S^1)^8 \subset \mathbb{R}^{16}$. We choose $\mathcal{T} = [0, 1]^{28}$ as there are 28 edges in K_8. We mix two types of random graphs with 8 vertices in training and testing: Erdős-Rényi graphs with $p = 0.5$ and K_8 with uniform edge weights in $[0, 1]$. Fig. 3 shows that POL can generate diverse set of max cuts. Quantitatively, compared to GOL, POL achieves better witnessed metrics (Fig. 5.D.10).

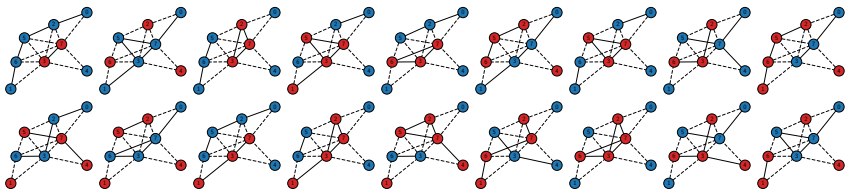

Figure 3: 18 different max cuts (max cut value 10) of a graph generated by our method. Red and blue vertices indicate the two vertex set separated by the cut. Vertex 0 is set to blue to remove the duplicates obtained by swapping the colors. See Fig. 5.D.12 for more results.

♢ 5.5.4 Symmetry detection of 3D shapes

Formulation. Geometric symmetries are omnipresent in natural and man-made objects. Knowing symmetries can benefit downstream tasks in geometry and vision [Mit+13; Shi+20; ZLM21]. We consider the problem of finding all reflection symmetries of a 3D surface. Let τ be a shape representation (e.g. point cloud, multi-view scan), and let $\mathcal{M}_\tau \subset \mathbb{R}^3$ denote the corresponding triangular mesh that is available for the training set. As reflections are determined by the reflectional plane, we set $\mathcal{X} = S^2 \times \mathbb{R}_{\geq 0}$, where $x = (n, d) \in \mathcal{X}$ denotes the plane with unit normal $n \in S^2 \subset \mathbb{R}^3$ and intercept $d \in \mathbb{R}_{\geq 0}$ (we assume $d \geq 0$ to remove the ambiguity of $(-n, -d)$ representing the same plane). Let $R_x : \mathbb{R}^3 \to \mathbb{R}^3$ denote the corresponding reflection. Perfect symmetries of $\mathcal{M}_\tau$ satisfy $R_x(\mathcal{M}_\tau) = \mathcal{M}_\tau$. Let $s_\tau : \mathbb{R}^3 \to \mathbb{R}$ be the (unsigned) distance field of $\mathcal{M}_\tau$ given by $s_\tau(p) = \min_{q \in \mathcal{M}_\tau} \|p - q\|_2$. Inspired by Podolak et al. [Pod+06], we define the MSO objective to be

$$f_\tau(x) \triangleq \mathbb{E}_{p \sim \mathcal{M}_\tau}[s_\tau(R_x(p))], \tag{5.8}$$

where a batch of p is sampled uniformly from $\mathcal{M}_\tau$ when evaluating the expectation. Although f_τ is stochastic, since we use point-to-mesh distances to compute s_τ, perfect symmetries will make (5.8) zero with probability one. Compared to existing methods that either require ground truth symmetries obtained by human annotators [Shi+20] or detect only a small number of symmetries [Gao+20], our method applied to (5.8) finds arbitrary numbers of symmetries including continuous ones and can generalize to unseen shapes, without needing ground truth symmetries as supervision.

Benchmark. We detect reflection symmetries for mechanical parts in the MCB dataset [Kim+20]. We choose $\mathcal{T}$ to be the space of 3D point clouds representing mechanical parts. From the mesh of each shape, we sample 2048 points with their normals uniformly and use DGCNN [Wan+19] to encode the oriented point clouds. Fig. 4 show our method's results on a selection of models in the test dataset; for per-iteration PPA results of our method, see Fig. 5.D.13. Fig. 5.D.11 shows that POL achieves much higher witnessed precision compared to GOL.

♢ 5.5.5 Object detection in images

Formulation. Identifying objects in an image is a central problem in vision on which recent works have made significant progress [Ren+15; Car+20; Liu+21]. We consider a simplified task where we drop the class labels and predict only bounding boxes. Let $b = (x, y, w, h) \in \mathcal{X} = [0, 1]^4$ denote a box with (normalized) center coordinates (x, y), width w, and height h. We choose $\mathcal{T}$ to be the space of images. Suppose an image τ has K_τ ground truth object bounding boxes $\{b_i^\tau\}_{i=1}^{K_\tau}$. We define the MSO objective to be

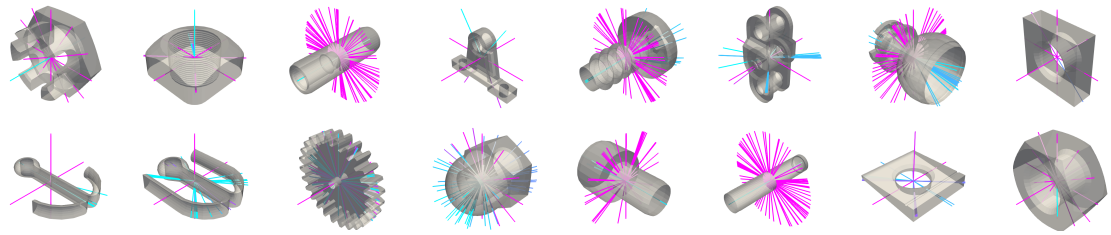

Figure 4: Symmetry detection results. Each reflection is represented as a colored line segment representing the normal of the reflection plane with one endpoint on the plane. Pink indicates better objective values, while blue indicates worse. Our method is capable of detecting complicated discrete symmetries as well as continuous families of cylindrical reflectional symmetries.

$f_\tau(x) \triangleq \min_{i=1}^{K_\tau} \|b_i^\tau - x\|_1$; its minimizers are exactly $\{b_i^\tau\}_{i=1}^{K_\tau}$. Although the objective may seem trivial, its gradients reveal the ℓ^1-Voronoi diagram formed by b_i^τ's when training the proximal operator. Different from existing approaches, we encode the distribution of bounding boxes conditioned on each image in the learned proximal operator without needing to predict confidence scores or a fixed number of boxes. A similar idea based on diffusion is recently proposed by Chen et al. [Che+23].

Benchmark. We apply the above MSO formulation to the COCO2017 dataset [Lin+14]. As τ is an image, we fine-tune ResNet-50 [He+16] to encode τ into a vector z that can be consumed by the operator network (Fig. 5.B.1).

Table 1: Object detection results. WD_∞ (resp. $\mathrm{WP}_\infty^{0.1}$) is the witnessed divergence (resp. precision) in (5.4) with $t = \infty$ (i.e. keeping all solutions), averaged over 10 trials (standard deviation $< 10^{-3}$). Precision and recall are computed with Hungarian matching as no confidence score is available for the usual greedy matching (see Sec. 5.D.8). FRCNN(.S) [Ren+15] means keeping predictions with confidence $\geq S\%$ for Faster R-CNN.

METHOD	WD_∞	$\mathrm{WP}_\infty^{0.1}$	PRECISION	RECALL
FRCNN(.80)	**0.140**	**0.624**	0.778	**0.650**
FRCNN(.95)	0.162	0.589	**0.887**	0.515
FN	0.161	0.481	0.139	**0.577**
GOL	0.251	0.243	0.508	0.282
POL (OURS)	**0.149**	**0.590**	**0.817**	0.442

In addition to GOL, we design a baseline method FN that uses the same ResNet-

50 backbone and predicts a fixed number of boxes using the chamfer distance as the training loss. Tab. 1 compares the proposed methods with alternatives and the highly-optimized Faster R-CNN [Ren+15] on the test dataset. Since we do not output confidence scores, the metrics are computed solely based on the set of predicted boxes. Our method achieves significantly better results than FN and GOL. Compared to the Faster R-CNN, we achieve slightly worse results with 40.7% fewer network parameters. While Faster R-CNN contains highly-specialized modules such as the regional proposal network, in our method we simply feed the image feature vector output by ResNet-50 to a general-purpose operator network. Incorporating specialized architectures like region proposal networks into our proximal operator learning framework for object detection is an exciting future direction. We visualize the effect of PPA using the learned proximal operator in Fig. 5. Further qualitative results (Fig. 5.D.14) and details can be found in Sec. 5.D.8.

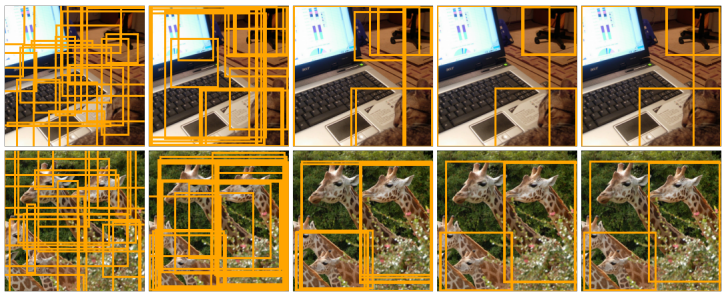

Figure 5: First 4 iterations of PPA using the learned proximal operator on 20 randomly initialized boxes (leftmost column). Only a few iterations are needed for the boxes to form distinctive clusters.

■ 5.6 Conclusion

Our work provides a straightforward and effective method to learn the proximal operator of MSO problems with varying parameters. Iterating the learned operator on randomly initialized points efficiently yields multiple optima to the MSO problems. Beyond promising results on our benchmark tasks, we see many exciting future directions that will further improve our pipeline.

A current limitation is that at test time the optimal number of iterations to apply the learned operator is not known ahead of time (see end of Sec. 5.D.1). One way to overcome this limitation would be to train another network that estimates when to stop. This measurement can be the objective itself if the optimum value is known *a priori* (e.g., sampling from level sets) or the gradient norm if objectives are smooth. One other future direction is to learn a proximal operator that adapts to multiple λ's. This way, the user

can easily experiment with different λ's and to enable PPA with growing step sizes for super-linear convergence [Roc76; Roc21]. Another direction is to study how much we can relax the assumption that $\mathcal{X}$ is a low-dimensional Euclidean space. Our method could remain effective when $\mathcal{X}$ is a low-dimensional submanifold of a high-dimensional Euclidean space. The challenges would be to constrain the proximal operator to a submanifold and to design a proximal term that is more suitable than the ambient ℓ^2 norm.

Appendices

■ 5.A Convergence of Training

We formally state and prove Thm. 5.3.1 via the following Prop. 5.A.1 and Prop. 5.A.2.

Proposition 5.A.1. *Suppose*

1. $\mathcal{T} \subset \mathbb{R}^r$ *for some* $r \in \mathbb{N}_{\geq 1}$;
2. *for any* $\tau \in \mathcal{T}$, *the objective* f_τ *is differentiable,* ξ*-weakly convex, and* ∇f_τ *is* ζ*-Lipschitz, i.e.,*
$$\|\nabla f_\tau(x_1) - \nabla f_\tau(x_2)\|_2 \leq \zeta \|x_1 - x_2\|_2,$$
with $\xi \leq \lambda$.
3. *the activation function* $\sigma(x)$ *used is proper, real analytic, monotonically increasing and 1-Lipschitz, e.g., sigmoid, hyperbolic tangent.*

For any $\delta > 0$, $H \geq 2$, $n \in \mathbb{N}_{\geq 1}$, *assume* Φ *is an* H*-layer feed-forward neural network with hidden layer sizes* $m_1, \ldots, m_H$ *satisfying*

$$m_1, \ldots, m_{H-2} \geq \Omega(H^2 \log(Hn^2/\delta)),$$
$$m_{H-1} \geq \Omega(\log(Hn^2/\delta)), \quad m_H \geq \Omega(n).$$

Let D *denote the total number of weights in* Φ. *Then* $D = \tilde{\Omega}(n)$. *Moreover, there exists a learning rate* $\eta \in \mathbb{R}^D$ *such that for any dataset* $S = \{(x_i, \tau_i)\}_{i=1}^n$ *of size* n *with the training loss* L *defined as in* (5.3), *for any* $\epsilon > 0$, *with probability at least* $1 - \delta$ *(over random Gaussian initial weights* θ^0 *of* Φ*), there exists* $t = O(c_r(\lambda + \zeta)/\epsilon)$ *such that* $L(\Phi(\cdot, \cdot; \theta^t)) \leq L^* + \epsilon$, *where* $\|\theta^t\|_2^2$ *stays bounded,* $L^* \triangleq \min_{g \in \mathcal{X} \times \mathcal{T} \to \mathcal{X}} L(g)$ *is the global minimum of the functional* L, $(\theta^k)_{k \in \mathbb{N}}$ *is the sequence generated by gradient descent* $\theta^{k+1} \triangleq \theta^k - \eta \odot \nabla_\theta L(\Phi(\cdot, \cdot; \theta^k))$, *and* c_r *depends only on* L *and the initialization* θ^0.

Proof of Prop. 5.A.1. The theorem is an application of Theorem 1 in Kawaguchi and Huang [KH19] with the following modifications.

For $i \in [n]$, define $\ell_i(x) \triangleq f_{\tau_i}(x) + \frac{\lambda}{2}\|x - x_i\|_2^2$. To check Assumption 1 of Kawaguchi and Huang [KH19], observe

$$\begin{aligned}\nabla_x \ell_i(x) &= \nabla f_{\tau_i}(x) + \lambda(x - x_i),\\ \nabla_x^2 \ell_i(x) &= \nabla^2 f_{\tau_i}(x) + \lambda I_d.\end{aligned}$$

Hence the assumption that f_{τ_i} is ξ-weakly convex implies that

$$\nabla^2 f_{\tau_i}(x) + \lambda I_d \succcurlyeq \nabla^2 f_{\tau_i}(x) + \xi I_d \succcurlyeq 0.$$

Hence ℓ_i is convex. The assumption that ∇f_{τ_i} is ζ-Lipschitz implies, for any $x_1, x_2 \in \mathcal{X} \times \mathcal{T}$,

$$\begin{aligned}\|\nabla \ell_i(x_1) - \nabla \ell_i(x_2)\|_2 &= \|\nabla f_{\tau_i}(x_1) - \nabla f_{\tau_i}(x_2) + \lambda(x_2 - x_1)\|_2\\ &\le \|\nabla f_{\tau_i}(x_1) - \nabla f_{\tau_i}(x_2)\|_2 + \lambda\|x_1 - x_2\|_2\\ &\le (\lambda + \zeta)\|x_1 - x_2\|_2.\end{aligned}$$

Hence $\nabla \ell_i$ is $(\lambda + \zeta)$-Lipschitz.

An input vector to the neural network Φ is the concatenation $(x, \tau) \in \mathbb{R}^{d+r}$. Kawaguchi and Huang [KH19] assume that the input data points are normalized to have unit length. This is not an issue, as we can scale down (x_i, τ_i) uniformly to be contained in a unit ball, then pad τ_i one extra coordinate to make $\|(x_i, \tau_i)\|_2 = 1$ for all $i \in [n]$, similar to the argument given in the footnotes before Assumption 2.1 of Allen-Zhu, Li, and Song [ALS19].

Lastly, we mention explicitly lower bounds for the layer sizes that are used in the proof of Theorem 1 of Kawaguchi and Huang [KH19] (see the paragraph below Lemma 3), instead of stating a single bound on the total number of weights in the statement of Theorem 1. This is because Theorem 1 only states that there *exists* a network of size $\tilde{\Omega}(n)$ for which training converges, whereas *every* network satisfying the layer-wise bounds will have the same convergence guarantee. □

Next we show that once the training loss is ϵ away from the global minimum, we can guarantee that the approximation error on the training data in the mean-squared sense is small: i.e., the learned operator $\Phi(\cdot, \cdot; \theta)$ is close to the true proximal operator (5.1).

Proposition 5.A.2. *Suppose for any $\tau \in \mathcal{T}$, the objective f_τ is differentiable and ξ-weakly convex with $\xi < \lambda$, where λ is the proximal regularization weight of the training loss $L(g)$ defined in (5.3). Let θ be the weight of the network Φ such that $L(\Phi(\cdot, \cdot; \theta)) \le L^* + \epsilon$ where $L^* \triangleq \min_{g \in \mathcal{X} \times \mathcal{T} \to \mathcal{X}} L(g)$ is the global minimum of the functional L. Let* $\mathrm{prox}(\cdot; \cdot)$ *be*

the true proximal operator defined in (5.1). Then the mean-squared error on the training data is bounded by

$$\frac{1}{n}\sum_{i=1}^{n}\|\Phi(x_i,\tau_i;\theta)-\operatorname{prox}(x_i;\tau_i)\|_2^2 \le \frac{2\epsilon}{\lambda-\xi}.$$

Proof. Clearly $L^* = L(\operatorname{prox}(\cdot;\cdot))$, i.e., the minimum of L is achieved with the true proximal operator. Define $h_i : \mathcal{X} \to \mathbb{R}$ by $h_i(x) \triangleq f_{\tau_i}(x) + \frac{\lambda}{2}\|x - x_i\|_2^2$, so that we can write $L(g) = \frac{1}{n}\sum_{i=1}^{n} h_i(g(x_i,\tau_i))$. By the assumption on weak convexity, each h_i is $(\lambda-\xi)$-strongly convex. This implies for any $x, y \in \mathcal{X}$,

$$h_i(x) \ge h_i(y) + \nabla h_i(y)^\top (x-y) + \frac{\lambda-\xi}{2}\|x-y\|_2^2. \tag{5.9}$$

The minimum of h_i is achieved at $\operatorname{prox}(x_i;\tau_i)$ by the definition of prox. Differentiability and convexity imply $\nabla h_i(\operatorname{prox}(x_i;\tau_i)) = 0$. Hence setting $y = \operatorname{prox}(x_i;\tau_i)$ in (5.9) implies, for any $x \in \mathcal{X}$,

$$h_i(x) - h_i(\operatorname{prox}(x_i;\tau_i)) \ge \frac{\lambda-\xi}{2}\|x-\operatorname{prox}(x_i;\tau_i)\|_2^2.$$

Now by the definition of (5.3),

$$\begin{aligned}
\epsilon &\ge L(\Phi(\cdot,\cdot;\theta)) - L^* = L(\Phi(\cdot,\cdot;\theta)) - L(\operatorname{prox}(\cdot,\cdot)) \\
&= \frac{1}{n}\sum_{i=1}^{n}\left[h_i(\Phi(x_i,\tau_i;\theta)) - h_i(\operatorname{prox}(x_i;\tau_i))\right] \\
&\ge \frac{1}{n}\sum_{i=1}^{n}\frac{\lambda-\xi}{2}\|\Phi(x_i,\tau_i;\theta)-\operatorname{prox}(x_i;\tau_i)\|_2^2.
\end{aligned}$$

Rearranging terms we obtain the desired result. □

■ 5.B Network Architectures

The network architecture we use to parameterize the operators for both POL and GOL is identical and is shown in Fig. 5.B.1. The encoder of τ will be chosen depending on the application. For our conic section (5.5.1), sparse recovery (5.5.2), and max-cut (5.5.3) benchmarks, the encoder is just the identity map. For symmetry detection (5.5.4), τ is a point cloud and we use DGCNN [Wan+19]. For object detection (5.5.5), we use ResNet-50 [He+16]. Inspired by Dinh, Sohl-Dickstein, and Bengio [DSB16], we include both additive and multiplicative coupling in the residual blocks. At the same time, since we do not need bijectivity of the operator (and proximal operators should not be) nor

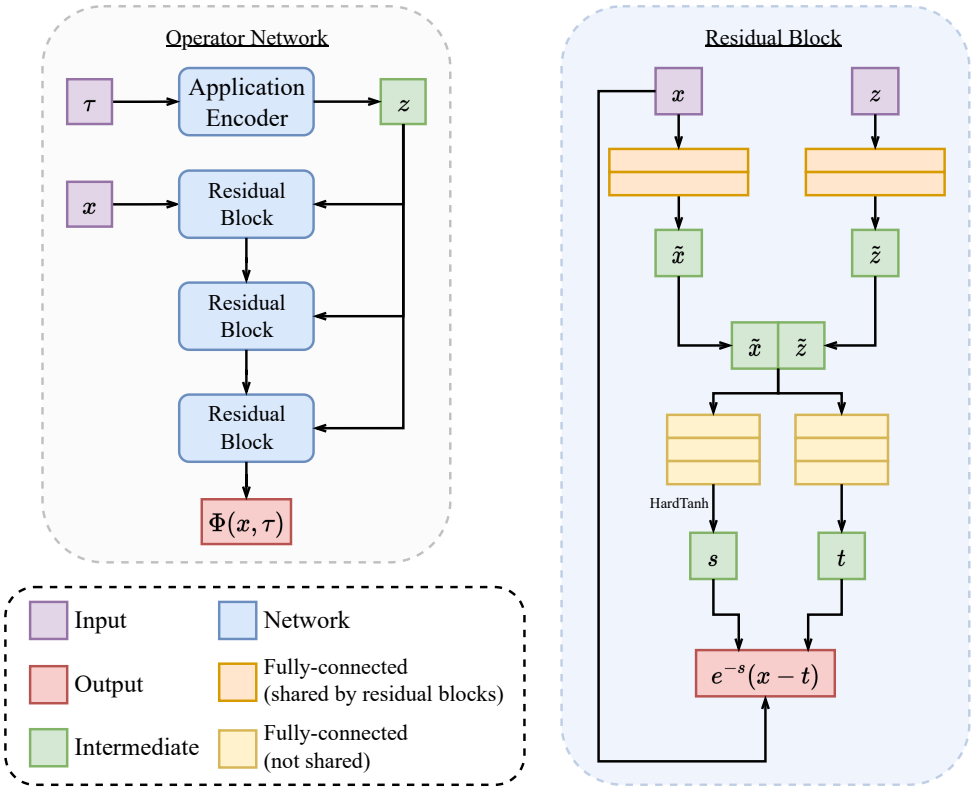

Figure 5.B.1: Network architecture for the operators used in POL and GOL. We use ReLU as the activation after all intermediate linear layers, except for predicting the scaling in the residual block, where we use HardTanh to ensure $s \in [-2, 2]$. For the shared 2-layer fully-connected network, the hidden layer sizes are $256, 128$. For the 3-layer fully-connected network in each residual block, the hidden layer sizes are $128, 128, 128$.

access to the determinant of the Jacobian, we do not restrict ourselves to a map with triangular structure as in Dinh, Sohl-Dickstein, and Bengio [DSB16]. We use 3 residual blocks for all applications, except for symmetry detection where we use 5 blocks which give slightly improved performance.

Our architecture is economical: the model size (excluding the application-specific encoder) is under 2MB for all applications we consider. This also makes iterating the operators fast at test time. Note that the application-specific encoder only needs to be run once at each test time τ as the encoded vector z can be reused (Fig. 5.B.1).

■ 5.C Importance Sampling via Unfolding PPA

Directly optimizing (5.2) or (5.5) using mini-batching may not yield an operator that can refine a near-optimal solution, if μ is taken to be Uniform($\mathcal{X}$), the uniform measure on $\mathcal{X}$ (more precisely, the d-dimensional Lebesgue measure restricted to $\mathcal{X}$ and normalized to a probability distribution). Instead, we would like to sample from a distribution that puts more probability density on near-optimal solutions. We achieve this goal as follows, inspired by Wang and Solomon [WS19]. Let Φ denote the network with weights after t training iterations. For $k \in \mathbb{N}_{\geq 0}$, denote $\mu^k \triangleq (\Phi^k)_{\#}(\text{Uniform}(\mathcal{X}))$. For a fixed $K \in \mathbb{N}_{\geq 1}$, we set $\mu \triangleq \frac{1}{K+1}\sum_{k=0}^{K} \mu^k$. Then, for training iteration $t+1$, we optimize the objective (5.2) or (5.5) with the constructed μ. Note this modification does not introduce any bias for POL (similarly for GOL), in the sense that the optimal solution to (5.2) is still the true proximal operator since μ has full support, yet it puts more density in near-optimal regions as t increases. In practice, we choose $K = 5$ or $K = 10$. For the choice of other hyper-parameters, see Sec. 5.D.1.

■ 5.D Detailed Results

◇ 5.D.1 Hyperparameters

Unless mentioned otherwise, the following hyper-parameters are used.

In each training iteration of POL and GOL, we sample 32 problem parameters from the training dataset of $\mathcal{T}$, and 256 of x's from Uniform($\mathcal{X}$) when computing (5.2) or (5.5) using the importance sampling trick in Sec. 5.C. The learning rate of the operator is kept at 10^{-4} for both POL and GOL, and by default we train the operator network for 2×10^5 iterations. This is sufficient for the loss to converge for both POL and GOL in most cases. Since GOL requires multiple evaluations of the gradient of the objective, it typically trains two or more times slower than POL. For the proximal weight λ of POL, we choose it based on the scale of the objective and the dimension of $\mathcal{X}$; see Tab. 5.D.1. All training is done on a single NVIDIA RTX 3090 GPU.

For the step size η in GOL, we start with $\frac{1}{\lambda}$ (so same step size as POL in the forward/backward Euler sense) and then slowly increase it (so fewer iterations are needed for convergence) without degrading the metrics. When evaluating (5.5), we set $Q = 10$ in all experiments except for symmetry detection, where we use $Q = 1$ because otherwise the training will take > 200 hours. For PD, we choose a step size small enough so as to not miss significant minima and a sufficient number of iterations for the loss (i.e. the objectives) to fully converge.

For evaluation, the number of iterations to apply the trained operators is chosen to be enough so that the objective converges. This number will be chosen separately for each application and method. By default, 1024 solutions are extracted from each method, and

Table 5.D.1: Choices of λ for all applications considered. $\mathcal{X}$ is the search space of solutions, and d is the dimension of the Euclidean space where we embed $\mathcal{X}$ (so it might be greater than the intrinsic dimension of $\mathcal{X}$).

Application	$\mathcal{X}$	d	λ
conic section (5.5.1)	$[-5,5]^2$	2	0.1
sparse recovery (5.5.2)	$[-2,2]^8$	8	10.0
max-cut (5.5.3)	$(S^1)^8$	16	10.0
symmetry detection (5.5.4)	$S^2 \times \mathbb{R}_{\geq 0}$	4	1.0
object detection (5.5.5)	$[0,1]^4$	4	1.0

1024 witnesses are sampled to compute WD_t and WP_t^δ, averaged over test dataset and over 10 trials with standard deviation provided (in most cases the standard deviation is two orders of magnitude smaller than the metrics). We filter out solutions that do not lie in $\mathcal{X}$.

A limitation for both POL and GOL is that when the solution set is continuous, too many applications of the learned operator can cause the solutions to collapse. We suspect this is because even with the importance sampling trick (Sec. 5.C), during training the operators may never see enough input that are near-optimal to learn the correct refinement needed to recover the continuous solution set. A future direction is to have another network to predict a confidence score for each $x \in \mathcal{X}$ so that at test time the user knows when to stop iterating the operator, e.g., when the objective value and its gradient are small enough; see the discussion in Sec. 5.6.

♢ 5.D.2 Convergence to the proximal operator

To empirically verify Prop. 5.A.2, that our method can faithfully approximate the true proximal operators of the objectives, we conduct the following simple experiments. We consider the function $f(x) = \|x\|_1$ for $x \in \mathcal{X} = [-1,1]^d$ and treat $\mathcal{T}$ as a singleton. Its proximal operator $\mathrm{prox}(x) = \arg\min_y \|y\|_1 + \|y - x\|_2^2$ is known in closed form as the shrinkage operation, defined coordinate-wise as:

$$\mathrm{prox}(x)_i = \begin{cases} x_i - 1/2 & x_i \geq 1/2 \\ 0 & |x_i| \leq 1/2 \\ x_i + 1/2 & x_i \leq -1/2. \end{cases} \tag{5.10}$$

For each dimension of $d = 2, 4, 8, 16, 32$, we train an operator network Φ (Fig. 5.B.1) using (5.2) as the loss with learning rate 10^{-3}. Fig. 5.D.1 shows the mean-squared-error $\|\Phi(x) - \Phi^*(x)\|_2^2$ scaled by $\frac{1}{d}$ and averaged over 1024 samples vs. the training iterations, where Φ^* is the shrinkage operation (5.10). We see that the trained operator indeed

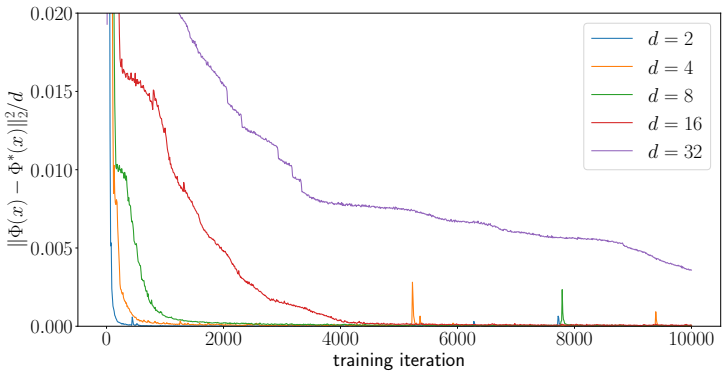

Figure 5.D.1: Convergence to the true proximal operator of $f(x) = \|x\|_1$.

converges to Φ^* as predicted by Prop. 5.A.2, and the convergence speed is faster in smaller dimensions.

♢ 5.D.3 Effect of the proximal term

In this section, we study the necessity of the proximal term $\frac{\lambda}{2}\|\Phi(x,\tau) - x\|^2$ in (5.2). Without such a term, the learned operator can degenerate. For example, consider (1) in [Che+22a], which minimizes $\min_\Phi \sum_{t=1}^T w_t f(x_t)$ with $x_{t+1} = x_t - \Phi(x_t, \nabla f(x_t), \ldots)$ for all t (with adapted notation). Suppose x^* is one global optimum of f but is not the only one. Then $\Phi(x, \ldots) := x - x^*$ clearly minimizes the objective, yet the update steps will always set $x_t = x^*$ regardless of the initial positions.

To further illustrate the effect of different choices of λ, consider the 2D cosine function $f(x) = -\sum_{i=1}^2 10\cos(2\pi x_i)$ for $x \in \mathcal{X} = [-5, 5]^2$ and a singleton $\mathcal{T}$. This function is ξ-weakly convex with $\xi = 40\pi^2 < 400$ and has global minima forming a grid (all local minima are global minima). On the left of Fig. 5.D.2, we see that when $\lambda = 400 > \xi$—in which case the condition of Thm. 5.3.1 is met—POL recovers all optima. In comparison, for $\lambda = 10$, the outer ring of solutions is missing, and with $\lambda = 0, 1$ most optima are missing in the grid.

To demonstrate how existing L2O methods can fail to recover multiple solutions, we conduct the same experiment on the L2O particle-swarm method by Cao et al. [Cao+19], which recovers a swarm of particles that are close to optima. We use the default parameters in the provided source code except changing the objective to the 2D cosine function and the standard deviation of the initial random particles to 1. As the method by Cao et al. [Cao+19] could produce particles outside $\mathcal{X} = [-5, 5]^2$, we add an additional term $0.01\|x\|^2$ to the objective $f(x)$; without such a term the particle swarm simply collapses to a single point far away from the origin. The results are shown on the right of Fig. 5.D.2. We see that even with 256 independent random starts and with population size 4, this

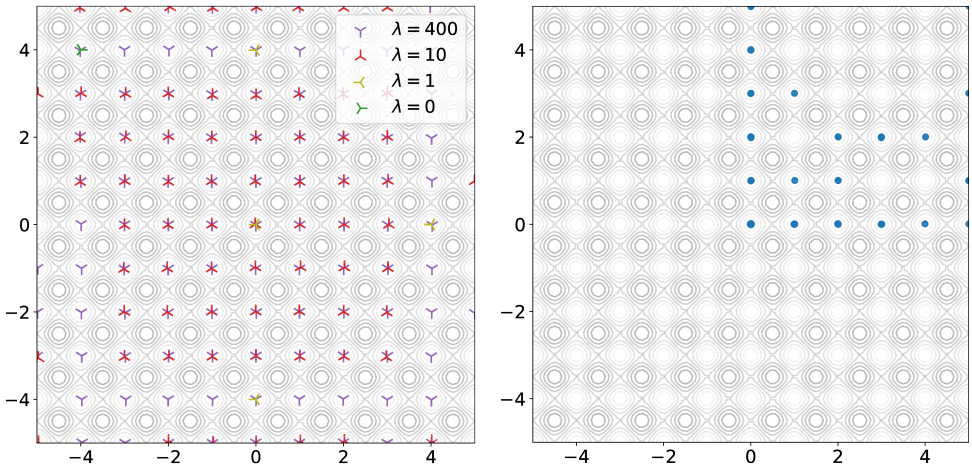

Figure 5.D.2: Left: the result of POL after 10 iterations with $\lambda = 0, 1, 10, 400$ for the 2D cosine function which has weakly convex constant $\xi = 40\pi^2 < 400$. Right: particle swarms recovered by Cao et al. [Cao+19] for after 10 iterations from 256 independent runs. The population size of the swarm is 4 (default value in their source code).

method fails to recover most of the optima, in particular in non-positive quadrants.

♢ 5.D.4 Sampling from conic sections

Setup. For this problem, the training dataset contains 2^{20} samples of $\tau \in \mathcal{T}$, while the test dataset has size 256. In our implementation and similarly in other benchmarks we do not store the dataset on disk, but instead generate them on the fly with fixed randomness. The τ's are sampled uniformly in $\mathcal{T}$. PD is run for 5×10^4 steps with learning rate 1.0. For step sizes, we choose $\lambda = 0.1$ for POL and $\eta = 1.0$ for GOL. We found that the training of GOL explodes when $\eta > 1.0$. Meanwhile, POL is able to take bigger ($\frac{1}{\lambda} = 10.0$) steps while staying stable during training (but might fail to recover solutions due to large step size). To obtain solutions, we use 5 iterations for POL, while for GOL we use 100 iterations since it converges slower (and more iterations won't improve the results).

Results. We visualize for the conic section problem in Fig. 5.D.3 for 16 randomly chosen $\tau \in \mathcal{T}$. In Fig. 5.D.4 we plot of δ vs. WP_t^δ (5.4) to quantitatively verify how good POL and GOL are at recovering the level sets, where we treat the results by PD as the ground truth. Both visually and quantitatively, we see that POL outperforms GOL. Fig. 5.D.5 compares the convergence speed when applying the learned iterative operators at test time: clearly POL converges much faster.

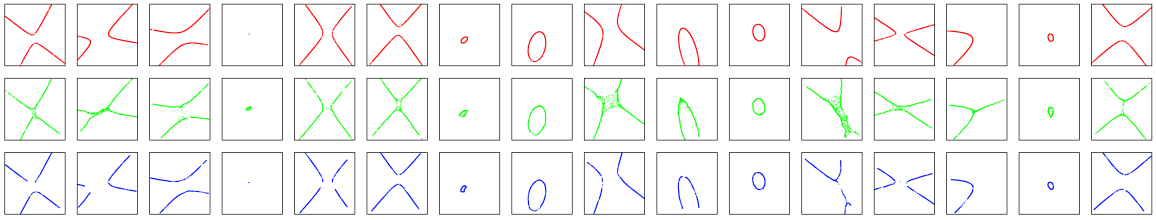

Figure 5.D.3: Visualization of the solutions for the conic section problem. Red indicates the solutions by PD which we treat as ground truth. Green and blue indicate the solutions by GOL (Sec. 5.4) and POL (proposed method) respectively.

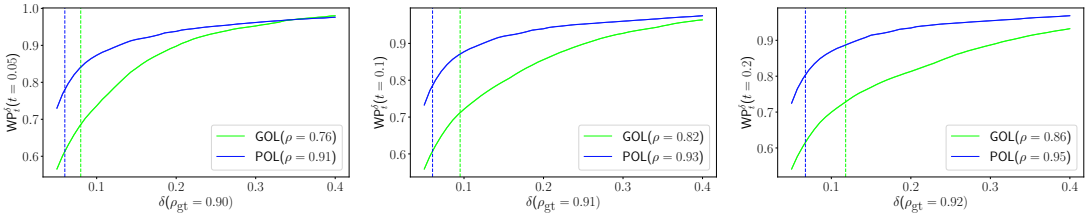

Figure 5.D.4: The plot of δ vs. WP_t^δ for the conic section problem ($t = 0.05, 0.1, 0.2$). The vertical dashed line indicates WD_t. 50 equally spaced δ values are used to draw the plot. Here ρ_{gt} indicates the percentage of PD solutions that have objectives $\leq t$, and ρ similarly indicates the percentage of solutions for each method with objectives below t. We sample 1024 witnesses to compute WP_t^δ, averaged over 256 test problem instances. The plot is averaged over 10 trials of witness sampling (the fill-in region's width indicates the standard deviation). Here the standard deviations are all less than 10^{-3} so the fill-in regions are too small to be visible.

♢ 5.D.5 Non-convex sparse recovery

Setup. For this problem, the training dataset contains 1024 samples of $\tau = (\alpha, p)$, while the test dataset has 128 samples. The τ's are sampled uniformly in $\mathcal{T} = [0, 1] \times [0.2, 0.5]$. We extract 4096 solutions from each method after training. For PD, we run 5×10^5 steps of gradient step with learning rate 10^{-5}. We found that due to the highly nonconvex landscape of the problem, bigger learning rates will cause PD to miss significant local minima. For step sizes, we choose $\lambda = 10$ for POL (so this corresponds to step size 0.1 for backward Euler) and $\eta = 0.1$ for GOL. To obtain solutions, POL requires less than 20 iterations to converge, while for GOL over 100 iterations are needed.

Results. We show the histogram of the solutions' objective values for PD, GOL, and POL in Fig. 5.D.7 for 4 problem instances. Fig. 5.D.6 visualizes the solutions for 8 problem instances projected onto the last two coordinates. GOL fails badly in all instances. Remarkably, despite the non-convexity of the problem and the much larger step size (0.1

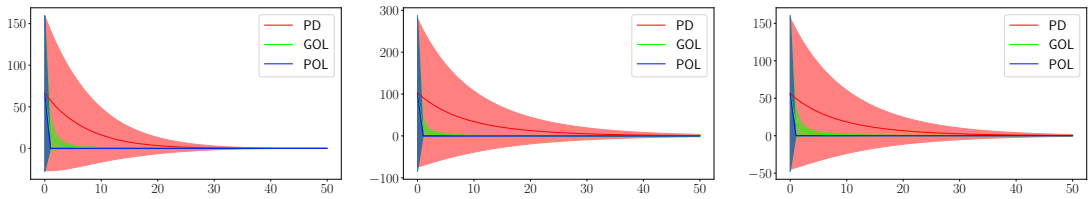

Figure 5.D.5: Convergence speed comparison at test time for the conic section problem. For POL and GOL, the x-axis is the number of iterations used. For PD, the x-axis is the number of gradient descent steps, multiplied by 100. The horizontal axis shows the number of iterations, and the vertical axis shows the value of $f_\tau(x)$, averaged over all current solutions (fill-in region's width indicates standard deviation). The three plots shown correspond to the problem instances in the first three columns in Fig. 5.D.3. Once the operator has been trained, POL converges in less than 5 steps, while GOL converges slower (GOL is already trained with the largest step size without causing training to explode).

compared to 10^{-5}), POL yields solutions on par or better than PD when p is small. For instance, for the second and third columns in Fig. 5.D.6 (corresponding to second and third columns in Fig. 5.D.7), PD (in red) misses near-optimal solutions that POL (in blue) captures. As such the results of PD can be suboptimal, so we do not compute witness metrics here.

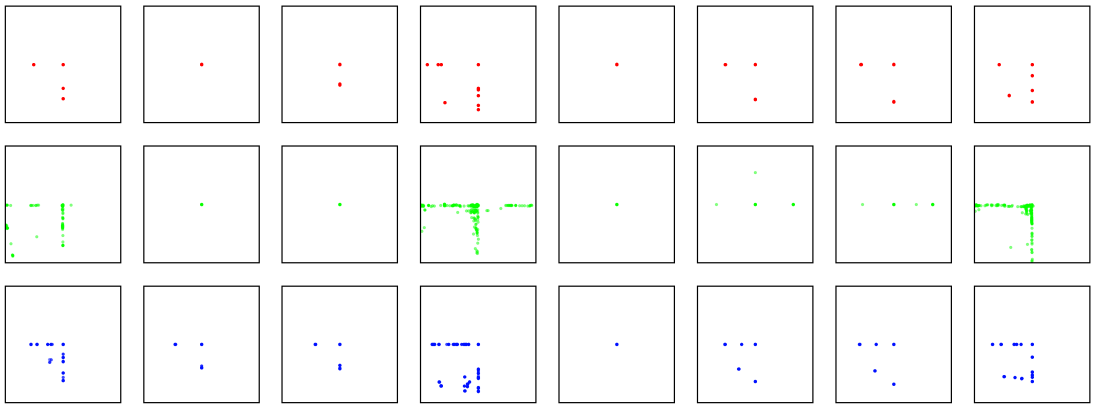

Figure 5.D.6: Visualization of the solutions' objective values for the conic section problem. Red indicates the solutions by PD which we treat as ground truth. Green and blue indicate the solutions by GOL and POL (proposed method) respectively.

Comparison with proximal gradient descent. For $p = \frac{1}{2}, \frac{2}{3}$, thresholding formulas exist for the ℓ^p norm [CSX13]. That is, the proximal operator of $\|x\|_p^p$ has a closed-form. This allows us to apply proximal gradient descent [Tib+10] to solve (5.6). When $p = 1$

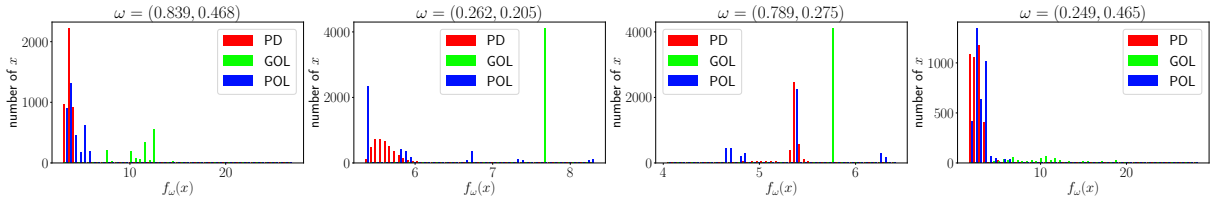

Figure 5.D.7: Histograms of the objectives for non-convex sparse recovery on 4 problem instances. We denote $\tau = (\alpha, p)$ at the top, corresponding to a sparsity-inducing function of the form $\alpha\|x\|_p^p$.

(i.e. when the problem reduces to LASSO), this reduces to the popular iterative soft-thresholding algorithm (ISTA) which converges significantly faster than gradient descent.

We compare the convergence speed of POL to that of proximal gradient descent (denoted PGD) for the $p = \frac{1}{2}$ case. We also include PD for reference. The generation of data (i.e. A and y in (5.6)) is the same as before. For POL we use the same setup with $\lambda = 10$ as before (corresponding to step size 0.1) except we restrict p to $\frac{1}{2}$ during training and we train only for 1000 steps (note the test-time α is unseen during training). For PGD we use 0.04 as the step size because using 0.05 for the step size would lead to divergence of PGD — the objective would go to infinity. For PD we use 0.05 as the step size. We run all three methods for 200 steps (for POL this corresponds to 200 steps of PPA after training) and visualize the convergence and histograms of the objective for each method in Fig. 5.D.8. We see that POL converges faster than PGD even when PGD is highly specialized to the $p = \frac{1}{2}$ case (where the thresholding formula has a closed form).

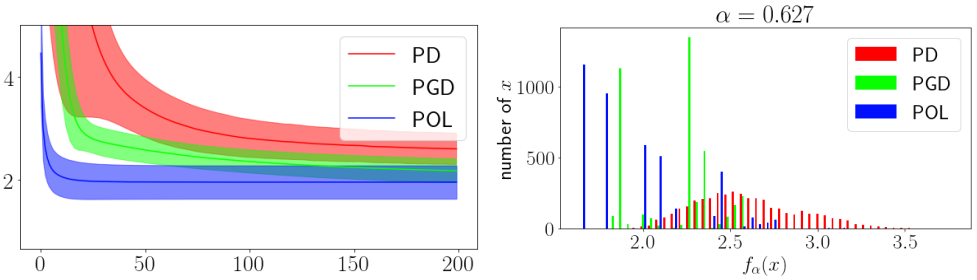

Figure 5.D.8: Convergence (left figure) and histograms (right figure) of the objective for the $p = \frac{1}{2}$ non-convex sparse recovery problem with $\alpha = 0.627$. For the convergence figure on the left, the horizontal axis shows the number of iterations, and the vertical axis shows the value of $f_\alpha(x)$, averaged over all current solutions (fill-in region's width indicates standard deviation). PGD denotes the proximal gradient descent method [Tib+10] with the closed-form thresholding formula by Cao, Sun, and Xu [CSX13].

Other sparsity-inducing regularizers. Our method can be applied to sparse re-

covery problems with other sparsity-inducing regularizers in a straightforward manner. Consider minimax concave penalty (MCP) from Yang et al. [Yan+20] defined componentwise as:

$$\mathrm{MCP}(x;\tau) \triangleq \begin{cases} |x| - \tau x^2 & |x| \le \frac{1}{2\tau} \\ \frac{1}{4\tau} & |x| > \frac{1}{2\tau}, \end{cases}$$

which is τ-weakly convex. We repeat the same setup as in Sec. 5.5.2 but with objectives

$$f_\tau(x) \triangleq \|Ax - y\|_2^2 + \sum_{i=1}^{d} \mathrm{MCP}(x_i;\tau), \tag{5.11}$$

for $\tau \in [0.5, 2]$. Note PGD is viable to solve (5.11) because the proximal operator of MCP has a closed-form. We run PGD for 2×10^4 iterations with step size 10^{-4} to make sure it converges fully. We show the histogram of the solutions' objective values for PD, GOL, POL, and PGD in Fig. 5.D.9. Our results are consistent with those in Fig. 5.D.7: POL is on par with PD and significantly outperforms GOL. POL also performs better than PGD which is only applicable because the regularizer MCP has a closed-form proximal operator.

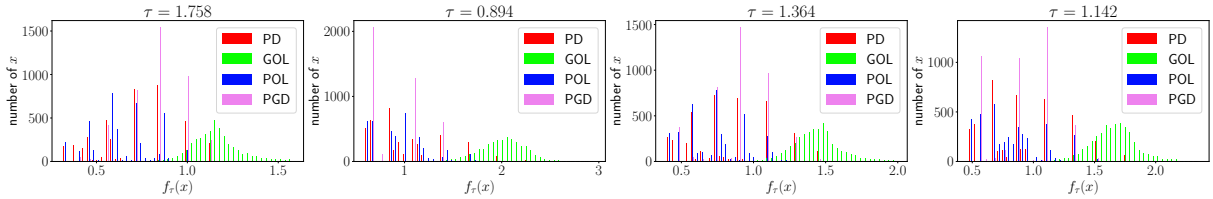

Figure 5.D.9: Histograms of the objectives for sparse recovery with minimax concave penalty (MCP) on 4 problem instances.

♢ 5.D.6 Rank-2 relaxation of max-cut

Setup. An additional feature of (5.7) is that the variables are constrained to $(S^1)^8 \subset \mathbb{R}^{16}$. Hence for POL and GOL we always project the output of the operator network to the constrained set (normalizing to unit length before computing the loss or before iterating), while for PD we apply projection after each gradient step.

We generate a training dataset of 2^{20} graphs and a test dataset of 1024 graphs using the procedure described in Sec. 5.5.3: half of the graphs will be Erdős-Rényi graphs with $p = 0.5$ and the remaining half being K_8 with edge weights drawn from $[0, 1]$ uniformly. For PD, we use learning rate 10^{-4}. For step sizes of POL and GOL, we choose $\lambda = 10.0$ and $\eta = 10.0$.

We choose to directly feed the edge weight vector $\tau \in \mathbb{R}^{28}$ to the operator network (Sec. 5.B). We find this simple encoding works better than alternatives such as graph

convolutional networks. This is likely because $x \in \mathcal{X} = (S^1)^8$ requires order information from the encoded τ, so graph pooling operation can be detrimental for the operator network architecture. Designing an equivariant operator network that is capable of effectively consuming larger graphs is an interesting direction for future work.

Results. If a cut happens to be a local minimum of the relaxation, then it is a maximum cut (Theorem 3.4 of Burer, Monteiro, and Zhang [BMZ02]). However, finding all the local minima of the relaxation is not enough to find all max cuts as max cuts can also appear as saddle points (see the discussion after Theorem 3.4 of Burer, Monteiro, and Zhang [BMZ02]). Hence solving the MSO (5.7) is not enough to identify all the max cuts. Nevertheless, we can still compare POL and GOL against PD based solely on the relaxed MSO problem corresponding to the objective (5.7). In Fig. 5.D.10, we plot δ vs. WP_t^δ

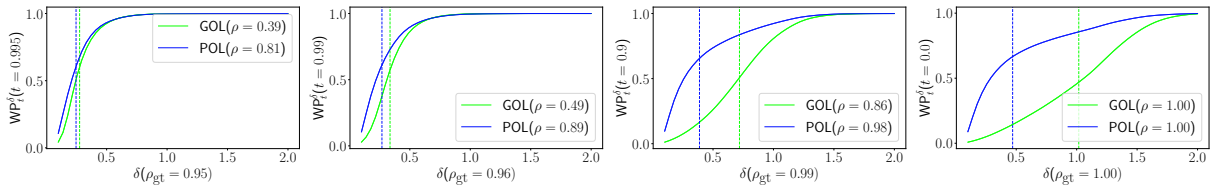

Figure 5.D.10: The plot of δ vs. WP_t^δ for the max-cut problem. See the caption of Fig. 5.D.4 for the meaning of the symbols. As different edge weights lead to different minima values, we choose the threshold t in a relative manner: the actual threshold used will be t times the best objective value found by PD.

(5.4) to verify the quality of the solutions obtained by POL and GOL compared to PD. We see that POL more faithfully recovers the solutions generated by PD with consistently higher witnessed precision.

Empirically, we found the proposed POL can identify a diverse family of cuts. We visualize the multiple cuts obtained by POL for a number of graphs in Fig. 5.D.12. Although some cuts are not maximal, they are likely due to the relaxation — not all fractional solutions correspond to a cut — and not because of the proposed method. As evident in Fig. 5.D.10, they are still very close to the local minima of (5.7) generated by PD.

♢ 5.D.7 Symmetry detection of 3D shapes

Setup. Since the variables in (5.8) is constrained to $\mathcal{X} = S^2 \times \mathbb{R}_{\geq 0}$, we always project the output of the operator network to the constrained set: for $x = (n, d) \in \mathcal{X}$, we normalize n to have unit length and take absolute value of d. The same projection is applied after each gradient step in PD.

To generate training and test datasets, we use the original train/test split of the MCB dataset [Kim+20] but filter out meshes with more than 5000 triangles and keep up to 100 meshes per category to make the categories more balanced. During each training

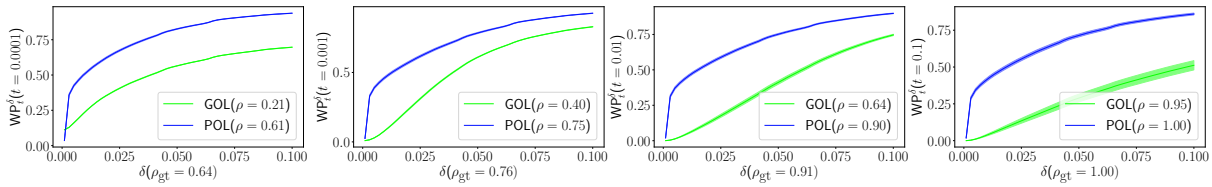

Figure 5.D.11: The plot of δ vs. WP_t^δ for symmetry detection on the test dataset of Kim et al. [Kim+20] ($t = 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}$). See the caption of Fig. 5.D.4 for the meaning of the various notations. We do not show WD_t as the vertical bars here because GOL's WD_t is much higher than POL's and is out of the range for the horizontal axis. We sample 1024 witnesses to compute WP_t^δ, averaged over 10 trials of witness sampling (the fill-in region's width indicates the standard deviation).

iteration, a fresh batch of point clouds are sampled (these are τ's) from the meshes in the current batch. For step sizes, we choose $\lambda = 1.0$ for POL and $\eta = 10.0$ for GOL. The training of POL and GOL takes about 30 hours. For PD, we run gradient descent for 500 iterations for each model, which is sufficient for convergence.

We use the official implementation of DGCNN by Wang et al. [Wan+19] as the encoder with the modification that we change the input channels to 6-dimension to consume oriented point clouds and we turn off the dropout layers which do not improve performance.

The objective (5.8) involves s_τ which requires point-to-mesh projection. We implemented custom CUDA functions to speed up the projection. Even so, it remains the bottleneck of training. Since GOL requires multiple evaluations, it is extremely slow and can take more than a week. As such, we set $Q = 1$ in (5.5). Both POL and GOL are trained for 10^5 iterations with batch size 8. At test time iterating the operator networks does not need to evaluate the objective nor the s_τ's; moreover, only point clouds are needed.

Results. We show the witness metrics in Fig. 5.D.11; quantitatively, POL exhibits far higher witnessed precision values than GOL. We show a visualization of iterations of PPA with the learned proximal operator in Fig. 5.D.13. In particular, our method is capable of detecting complicated discrete reflectional symmetries as well as a continuous family of reflectional symmetries for cylindrical objects.

♢ 5.D.8 Object detection in images

Setup. We use the training and validation split of COCO2017 [Lin+14] as the training and test dataset, keeping only images with at most 10 ground truth bounding boxes. For training, we use common augmentation techniques such as random resize/crop, horizontal flip, and random RGB shift, to generate a 400×400 patch from each training batch image, with batch size 32. For evaluation, we crop a 400×400 image patch from each test image. For step sizes, we choose $\lambda = 1.0$ for POL and $\eta = 1.0$ for GOL. We train both POL and

GOL for 10^6 steps. This takes about 100 hours. To extract solutions, we use 100 iterations for POL (for most images it only needs 5 iterations to converge) and 1000 iterations for GOL (the convergence is very slow so we run it for a large number of iterations).

We fine-tune PyTorch's pretrained ResNet-50 [He+16] with the following modifications. We first delete the last fully-connected layer. Then we add an additional linear layer to turn the 2048 channels into 256. We then add sinusoidal positional encodings to pixels in the feature image output by ResNet-50 followed by a fully-connected layers with hidden layer sizes $256, 256, 256$. Finally average pooling is used to obtain a single feature vector for the image.

For Faster R-CNN (FRCNN), we use the pretrained model from PyTorch with ResNet-50 backbone and a regional proposal network. It should be noted that FRCNN is designed for a different task that includes prediction of class labels, and thus it is trained with more supervision (object class labels) than our method and it uses additional loss terms for class labels.

For the alternative method FN that predicts a fixed number of boxes, we attach a fully-connected layer of hidden sizes $[256, 256, 80]$ with ReLU activation to consume the pooled feature vector from ResNet-50. The output vector of dimension 80 is then reshaped to 20×4, representing the box parameters of 20 boxes. We use chamfer distance between the set of predicted boxes and the set of ground truth boxes as the training loss.

Results. In Tab. 1, we compute witness metrics and traditional metrics including precision and recall. As our method does not output confidence scores, we cannot use common evaluation metrics such as average precision. To calculate precision and recall, which normally would require an order given by the confidence scores, we instead build a bipartite graph between the predicted boxes and the ground truth, adding an edge if the Intersection over Union (IoU) between two boxes is greater than 0.5. Then we consider predictions that appear in the Hungarian max matching as true positives, and the unmatched ones false positives. Then precision is defined as the number of true positives over the total number of predictions, while recall is defined as the number of true positives over the total number of ground truth boxes. When computing metrics for POL and GOL, we run mean-shift algorithm with RBF bandwidth 0.01 to find the centers of clusters and use them as the predictions. As shown in Fig. 5, the clusters formed by POL are usually extremely sharp after a few steps, and any reasonable bandwidth will result in the same clusters.

In Fig. 5.D.14, we show the detection results by our method for a large number of test images chosen at random.

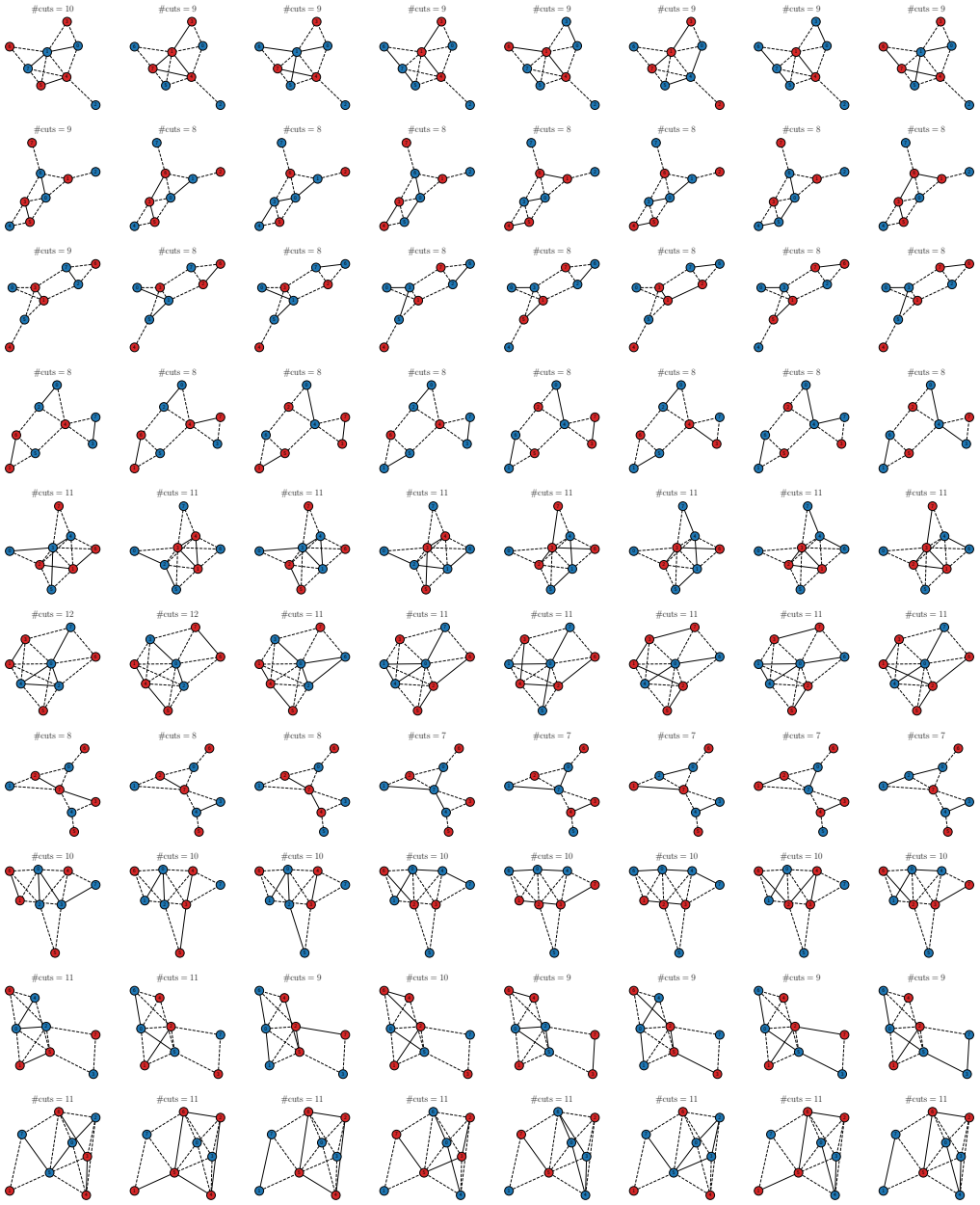

Figure 5.D.12: Visualization of the cuts obtained by applying Goemans-Williamson-type procedure to randomly selected solutions of POL. The graphs are chosen among the ones that have at least 8 solutions uniformly at random without human intervention. Each row contains the multiple solutions for the same graph with binary weights. Two colors indicate the two vertex sets separated by the cut. Dashed lines indicate edges in the cut. For each graph we annotate the number of cuts on top. The 0th vertex is always in blue to remove the obvious duplicates obtained by swapping the two colors on each vertex.

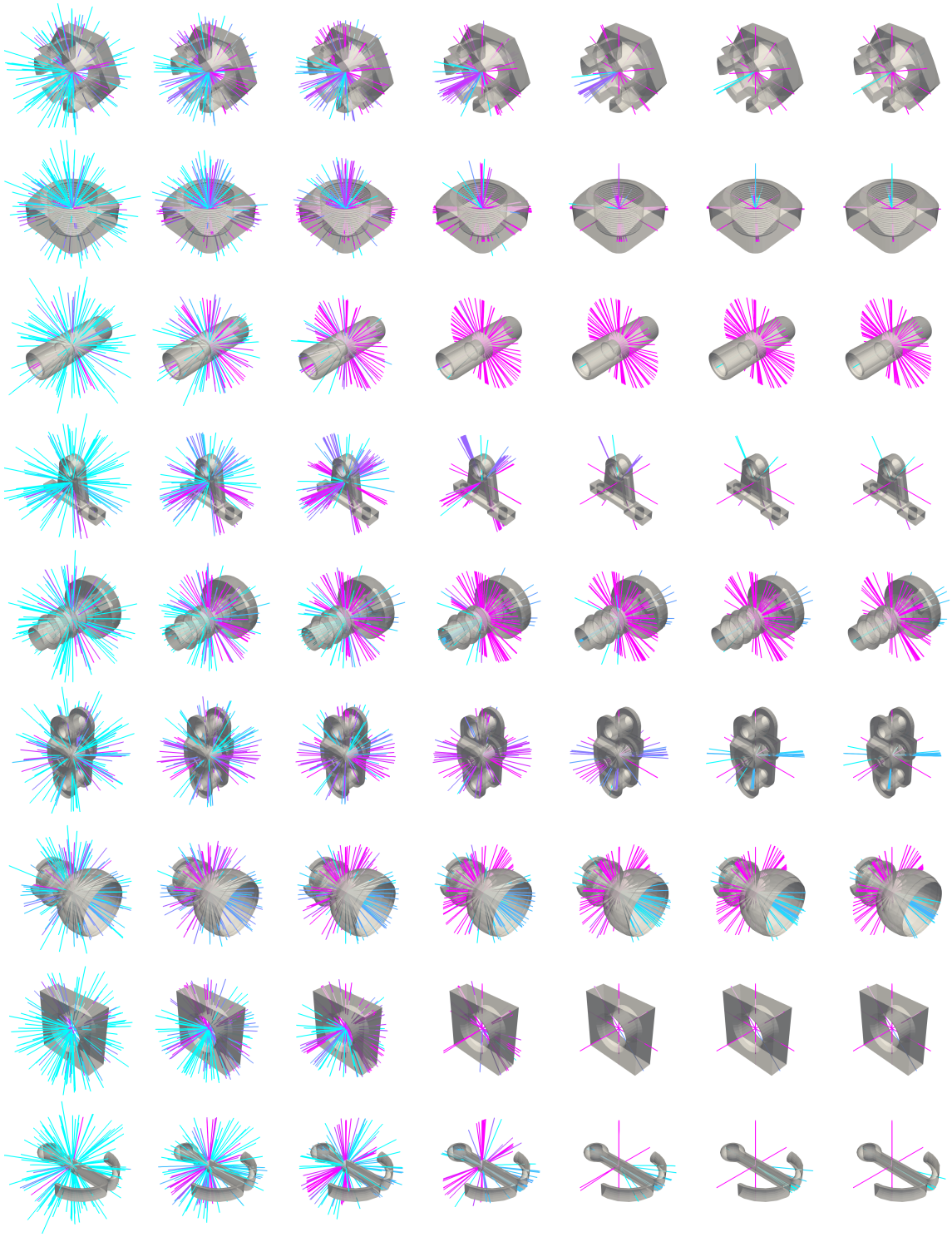

Figure 5.D.13: Visualization of PPA with learned proximal operator on selected models from the test dataset of Kim et al. [Kim+20]. Iterations 0, 1, 2, 5, 10, 15, 20 are shown, where the 0th iteration contains the initial samples from Uniform($\mathcal{X}$). Pink indicates lower objective value in (5.8), while light blue indicates higher.

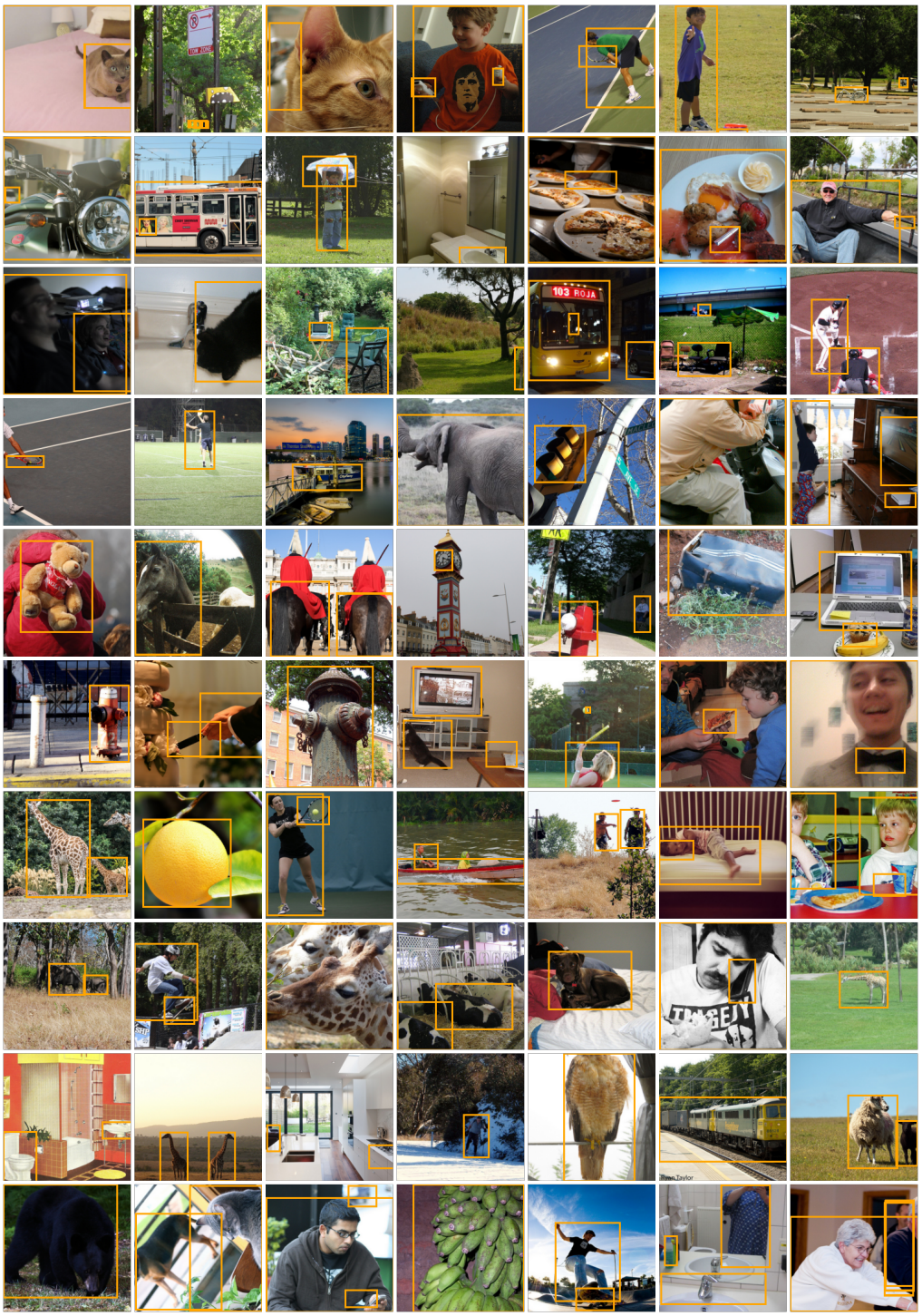

Figure 5.D.14: Randomly selected object detection results by POL on COCO17 validation split. Each test image is 400×400 patch cropped from the original image with a random scaling by a number between 0.5 and 1. In each image we display all 1024 bounding boxes without clustering, most of which are perfectly overlapping. For some images there is a bounding box for the whole image (check if the image has an orange border). There is no class label associated with each box.

■ 5.E Connection to Wasserstein Gradient Flows

In this section we show that we can view (5.2) as solving the JKO discretization of Wasserstein gradient flows at every time step, under the assumption that the measures along the JKO discretization are absolutely continuous.

If $\mathcal{F}(\mu)$ is a linear functional of the form $\mathcal{F}(\mu) = \int f \mathrm{d}\mu$ on $\mathcal{P}(\mathcal{X})$, the space of probability distributions in $\mathcal{X}$ with $\mathcal{X}$ compact, then the JKO discretization of the gradient flow of $\mathcal{F}$ at step $t+1$ with step size $1/\lambda$ is

$$\mu_{t+1} = \operatorname*{arg\,min}_{\mu \in \mathcal{P}_2(\mathcal{X})} \left\{ \int f \mathrm{d}\mu + \frac{\lambda}{2} W_2^2(\mu_t, \mu) \right\},$$

where W_2 is the Wasserstein-2 distance and we assume μ_t is absolutely continuous. Let

$$\mathcal{F}(\mu) \triangleq \int f \mathrm{d}\mu + \frac{\lambda}{2} W_2^2(\mu_t, \mu).$$

Let us also define another functional such that for a Borel map $T : \mathcal{X} \to \mathcal{X}$,

$$\mathcal{G}(T) \triangleq \int f(T(x)) \mathrm{d}\mu_t(x) + \frac{\lambda}{2} \int \|T(x) - x\|_2^2 \mathrm{d}\mu_t(x).$$

First given $\mu \in \mathcal{P}(\mathcal{X})$, since $\mathcal{X}$ is compact (so in particular all probability distributions have finite second moments), by Brenier's theorem [AGS05, Theorem 6.2.4], there exists a Borel map T (the *Monge map*) such that $T_{\#}\mu_t = \mu$ and $W_2^2(\mu_t, \mu) = \int \|T(x) - x\|_2^2 \mathrm{d}\mu_t(x)$. Hence for such μ and T we have $\mathcal{G}(T) = \mathcal{F}(\mu)$, and thus $\min_{\mu \in \mathcal{P}_2(\mathbb{R}^d)} \mathcal{F}(\mu) \geq \min_T \mathcal{G}(T)$.

Next given a Borel T, let $\mu = T_{\#}\mu_t$. By Brenier's theorem, let T' be the Monge map corresponding to $W_2(\mu_t, \mu)$ so that $\mu = T'_{\#}\mu_t$ and $\int \|T'(x) - x\|_2^2 \mathrm{d}\mu_t(x) = W_2(\mu_t, \mu) \leq \int \|T(x) - x\|_2^2 \mathrm{d}\mu_t(x)$. This shows that $\mathcal{F}(\mu) \leq \mathcal{G}(T)$ and hence $\min_{\mu \in \mathcal{P}_2(\mathbb{R}^d)} \mathcal{F}(\mu) \leq \min_T \mathcal{G}(T)$. Thus $\min_{\mu \in \mathcal{P}_2(\mathbb{R}^d)} \mathcal{F}(\mu) = \min_T \mathcal{G}(T)$.

If μ_t has full support, then the best $T^* \triangleq \arg\min_T \mathcal{G}(T)$ is obtained pointwise and it becomes the proximal operator of f (cf. (5.2)). In particular, T^* does not depend on μ_t.

Chapter 6

Self-Consistent Velocity Matching of Probability Flows

In Chapters 4 and 5, we demonstrated that neural networks can effectively parameterize continuous distributions, achieving strong performance. In this chapter, we optimize for a *probability flow* $(\mu_t)_t$ where each μ_t is a probability distribution, and moreover, the probability flow is jointly continuous in time and space as dictated by the continuity equation. Specifically, we exploit the self-consistency principle to solve a wide class of partial differential equations (PDEs) using neural network paramterization without spatial or temporal discretization. For optimization, we introduce an iterative method that solves a convex least square problem at each iteration, circumventing the significant computational challenges encountered in prior methods. This chapter is based on the publication [LHS23].

■ 6.1 Introduction

Mass conservation is a ubiquitous phenomenon in dynamical systems arising from fluid dynamics, electromagnetism, thermodynamics, and stochastic processes. Mathematically, mass conservation is formulated as the *continuity equation*:

$$\partial_t p_t(x) = -\nabla \cdot (v_t p_t), \forall x, t \in [0, T], \tag{6.1}$$

where $p_t : \mathbb{R}^d \to \mathbb{R}$ is a scalar quantity such that the total mass $\int p_t(x)$ is conserved with respect to t, $v_t : \mathbb{R}^d \to \mathbb{R}^d$ is a velocity field, and $T > 0$ is total time. We will assume, for all $t \in [0, T]$, $p_t \geq 0$ and $\int p_t(x)\mathrm{d}x = 1$, i.e., p_t is a probability density function. We use μ_t to denote the probability measure with density p_t. Once a pair (p_t, v_t) satisfies (6.1), the density p_t is coupled with v_t in the sense that the evolution of p_t in time is characterized by v_t (Sec. 6.3.1).

We consider the subclass of mass-conserving PDEs that can be written succinctly as

$$\partial_t p_t(x) = -\nabla \cdot (f_t(x; \mu_t) p_t), \forall x, t \in [0, T], \tag{6.2}$$

where $f_t(\cdot; \mu_t) : \mathbb{R}^d \to \mathbb{R}^d$ is a given function depending on μ_t, with initial condition $\mu_0 = \mu_0^*$ for a given initial probability measure μ_0^* with density p_0^*.

Different choices of f_t lead to a large class of mass-conserving PDEs. For instance, given a functional $\mathcal{F} : \mathcal{P}_2(\mathbb{R}^d) \to \mathbb{R}$ on the space of probability distributions with finite second moments, if we take

$$f_t(x; \mu_t) \triangleq -\nabla_{W_2} \mathcal{F}(\mu_t)(x), \tag{6.3}$$

where $\nabla_{W_2} \mathcal{F}(\mu) : \mathbb{R}^d \to \mathbb{R}^d$ is the Wasserstein gradient of $\mathcal{F}$, then the solution to (6.2) is the Wasserstein gradient flow of $\mathcal{F}$ [San15, Chapter 8]. Thus, solving (6.2) efficiently allows us to optimize in the probability measure space. If we take

$$f_t(x; \mu_t) \triangleq b_t(x) - D_t(x) \nabla \log p_t(x), \tag{6.4}$$

where b_t is a velocity field and $D_t(x)$ is a positive-semidefinite matrix, then we obtain the time-dependent Fokker-Planck equation [RR96], which describes the time evolution of the probability flow undergoing drift b_t and diffusion with coefficient D_t.

A popular strategy to solve (6.2) is to use an Eulerian representation of the density field p_t on a discretized mesh or as a neural network [RPK19]. However, these approaches do not fully exploit the mass-conservation principle and are usually limited to low dimensions. Shen et al. [She+22] and Shen and Wang [SW24] recently introduced the notion of *self-consistency* for the Fokker-Planck equation and more generally McKean-Vlasov type PDEs. This notion is a Lagrangian formulation of (6.2). They apply the adjoint method to optimize self-consistency. In this work, we extend their notion of self-consistency to mass-conserving PDEs of the general form (6.2). Equipped with this formulation, we develop an iterative optimization scheme called *self-consistent velocity matching*. With the probability flow parameterized as a neural network, at each iteration, we refine the velocity field v_t of the current flow to match an estimate of f_t evaluated using the network weights from the previous iteration. Effectively, we minimize the self-consistency loss with a biased but more tractable gradient estimator.

This simple scheme has many benefits. First, the algorithm is agnostic to the form of f_t, thus covering a wider range of PDEs compared to past methods. Second, we no longer need to differentiate through differential equations using the adjoint method as in Shen and Wang [SW24], which is orders of magnitude slower than our method with worse performance in high dimensions. Third, this iterative formulation allows us to rewrite the velocity-matching objectives for certain PDEs to get rid of computationally expensive quantities such as $\nabla \log p_t$ in the Fokker-Planck equation (Prop. 6.3.1). Lastly,

our method is flexible with probability flow parameterization: we have empirically found that the two popular ways of parameterizing the flow—as a time-varying pushforward map [Bil+21] and as a time-varying velocity field [Che+18]—both have merits in different scenarios.

Our method tackles mass-conserving PDEs of the form (6.2) in a unified manner without temporal or spatial discretization. Despite using a biased gradient estimator, in practice, our method decreases the self-consistency loss efficiently (second column of Fig. 2). For PDEs with analytically-known solutions, we quantitatively compare with the recent neural JKO-based methods [Mok+21; FTC21; ASM21], the adjoint method [SW24], and the particle-based method [BV23]. Our method faithfully recovers true solutions with quality on par with the best previous methods in low dimensions and with superior quality in high dimensions. Our method is also significantly faster than competing methods, especially in high dimensions, at the same time without discretization. We further demonstrate the flexibility of our method on two challenging experiments for modeling flows splashing against obstacles and smooth interpolation of measures where the comparing methods are either not applicable or have noticeable artifacts.

■ 6.2 Related Works

Classical PDE solvers for mass-conserving PDEs such as the Fokker-Planck equation and the Wasserstein gradient flow either use an Eulerian representation of the density and discretize space as a grid or mesh [BCW10; CCH15; Pey15] or use a Lagrangian representation, which discretizes the flow as a collection of interacting particles simulated forward in time [CL99; WW10]. Due to spatial discretization, these methods struggle with high-dimensional problems. Hence, the rest of the section focuses solely on recent neural network-based methods.

Physics-informed neural networks. Physics-informed neural networks (PINNs) are prominent methods that solve PDEs using deep learning [RPK19; Kar+21]. The main idea is to minimize the residual of the PDE along with loss terms to enforce the boundary conditions and to match observed data. Our notion of self-consistency is a Lagrangian analog of the residual in PINN. Our velocity matching only occurs along the flow of the current solution where interesting dynamics happen, while in PINNs the residual is evaluated on collocation points that occupy the entire domain. Hence our method is particularly suitable for high-dimensional problems where the dynamics have a low-dimensional structure.

Neural JKO methods. Recent works [Mok+21; ASM21; FTC21] apply deep learning to the time-discretized JKO scheme [JKO98] to solve Wasserstein gradient flow (6.3). By

pushing a reference measure through a chain of neural networks parameterized as input-convex neural networks (ICNNs) [AXK17], these methods avoid discretizing the space. Mokrov et al. [Mok+21] optimize one ICNN to minimize Kullback-Leibler (KL) divergence plus a Wasserstein-2 distance term at each JKO step. This method is extended to other functionals by Alvarez-Melis, Schiff, and Mroueh [ASM21]. Fan, Taghvaei, and Chen [FTC21] use the variational formulation of f-divergence to obtain a faster primal-dual approach.

An often overlooked problem of neural JKO methods is that the total training time scales quadratically with the number of JKO steps: to draw samples for the current step, initial samples from the reference measure must be passed through a long chain of neural networks, along with expensive quantities like densities. However, using too few JKO steps results in large temporal discretization errors. Moreover, the optimization at each step might not have fully converged before the next step begins, resulting in an unpredictable accumulation of errors. In contrast, our method does not suffer from temporal discretization and can be trained end-to-end. It outperforms these neural JKO methods with less training time in experiments we considered.

Velocity matching. A few recent papers employ the idea of velocity matching to construct a flow that follows a learned velocity field. Langosco, Fortuin, and Strathmann [LFS21] simulate the Wasserstein gradient flow of the KL divergence by learning a velocity field that drives a set of particles forward in time for Bayesian posterior inference. The velocity field is refined on the fly based on the current positions of the particles. Boffi and Vanden-Eijnden [BV23] propose a similar method that applies to a more general class of time-dependent Fokker-Planck equations. These two methods approximate probability measures using finite particles which might not capture high-dimensional distributions well. Liu, Gong, and Liu [LGL22], Lipman et al. [Lip+22], and Albergo and Vanden-Eijnden [AV22] use flow matching for generative modeling by learning a velocity field that generates a probability path connecting a reference distribution to the data distribution. Yet these methods are not designed for solving PDEs.

Most relevant to our work, Shen et al. [She+22] propose the concept of self-consistency for the Fokker-Planck equation, later extended to McKean-Vlasov type PDEs [SW24]. They observe that the velocity field of the flow solution to the Fokker-Planck equation must satisfy a fixed-point equation. They theoretically show that, under certain regularity conditions, a form of probability divergence between the current solution and the true solution is bounded by the self-consistency loss that measures the violation of the fixed-point equation. Their algorithm minimizes such violation using neural ODE parameterization [Che+18] and the adjoint method. Our work extends the concept of self-consistency to a wider class of PDEs in the form of (6.2) and circumvents the computationally demanding adjoint method using an iterative formulation. We empirically verify that our method

is significantly faster and reduces the self-consistency loss more effectively in moderate dimensions than that of Shen and Wang [SW24] (Fig. 2).

■ 6.3 Self-Consistent Velocity Matching

♢ 6.3.1 Probability flow of the continuity equation

A key property of the continuity equation (6.1) is that any solution $(p_t, v_t)_{t\in[0,T]}$ (provided p_t is continuous with respect to t and v_t is bounded) corresponds to a unique flow map $\{\Phi_t(\cdot) : \mathbb{R}^d \to \mathbb{R}^d\}_{t\in[0,T]}$ that solves the ordinary differential equations (ODEs) [AGS05, Proposition 8.1.8]

$$\Phi_0(x) = x, \frac{\mathrm{d}}{\mathrm{d}t}\Phi_t(x) = v_t(\Phi_t(x)), \forall x, t \in [0, T], \tag{6.5}$$

and the flow map satisfies $\mu_t = (\Phi_t)_{\#}\mu_0$ for all $t \in [0, T]$, where $(\Phi_t)_{\#}\mu_0$ to denote the pushforward measure of μ_0 by Φ_t. Moreover, the converse is true: any solution (Φ_t, v_t) of (6.5) with Lipschitz continuous and bounded v_t is a solution of (6.1) with $\mu_t = (\Phi_t)_{\#}\mu_0$ [AGS05, Lemma 8.1.6]. Thus the Eulerian viewpoint of (6.1) is equivalent to the Lagrangian viewpoint of (6.5). We next exploit this equivalence by modeling the probability flow using the Lagrangian viewpoint so that it automatically satisfies the continuity equation (6.1).

♢ 6.3.2 Parametrizing probability flows

Our algorithm will be agnostic to the exact parameterization used to represent the probability flow. As such, we need a way to parameterize the flow to access the following quantities for all $t \in [0, T]$:

- $\Phi_t : \mathbb{R}^d \to \mathbb{R}^d$, the flow map, so $\Phi_t(x)$ is the location of a particle at time t if it is at x at time 0.
- $v_t : \mathbb{R}^d \to \mathbb{R}^d$, the velocity field of the flow at time t.
- $\mu_t \in \mathcal{P}(\mathbb{R}^d)$, the probability measure at time t with sample access and density p_t.

We will assume all these quantities are sufficiently continuous and bounded to ensure the Eulerian and Lagrangian viewpoints in Sec. 6.3.1 are equivalent. This can be achieved by using continuously differentiable activation functions in the network architectures and assuming the network weights are finite similar to the uniqueness arguments given in [Che+18]. We can now parameterize the flow modeling either the flow map Φ_t or the velocity field v_t as a neural network.

Time-dependent Invertible Push Forward (TIPF). We first parameterize a probability flow by modeling $\Phi_t : \mathbb{R}^d \to \mathbb{R}^d$ as an invertible network for every t. The network architecture is chosen so that Φ_t has an analytical inverse with a tractable Jacobian determinant, similar to [Bil+21]. We augment RealNVP from [DSB16] so that the network for predicting scale and translation takes t as an additional input. To enforce the initial condition, we need Φ_0 to be the identity map. This condition can be baked into the network architecture [Bil+21] or enforced by adding an additional loss term $\mathbb{E}_{X\sim\mu_0^*}\|\Phi_0(X) - X\|^2$. For brevity, we will from now on omit in the text this additional loss term. The velocity field can be recovered via $v_t(x) = \partial_t\Phi_t(\Phi_t^{-1}(x))$. To recover the density p_t of $\mu_t = (\Phi_t)_\#\mu_0$, we use the change-of-variable formula $\log p_t(x) = \log p_0^*(\Phi_t^{-1}(x)) + \log\det\left|J\Phi_t^{-1}(x)\right|$ (see (1) in Dinh, Sohl-Dickstein, and Bengio [DSB16]).

Neural ODE (NODE). We also parameterize a flow by modeling $v_t : \mathbb{R}^d \to \mathbb{R}^d$ as a neural network; this is used in Neural ODE [Che+18]. The network only needs to satisfy the minimum requirement of being continuous. The flow map and the density can be recovered via numerical integration: $\Phi_t(x) = x + \int_0^t v_s(\Phi_s(x))\mathrm{d}s$ and $\log p_t(\Phi_t(x)) = \log p_0^*(x) - \int_0^t \nabla\cdot v_s(\Phi_s(x))\mathrm{d}s$, a direct consequence of (6.1) also known as the instantaneous change-of-variable formula [Che+18]. To obtain the inverse of the flow map, we integrate along $-v_t$. With NODE, the initial condition $\mu_0 = \mu_0^*$ is obtained for free.

We summarize the advantages and disadvantages of TIPF and NODE as follows. While the use of invertible coupling layers in TIPF allows exact access to samples and densities, TIPF becomes less effective in higher dimensions as many couple layers are needed to retain good expressive power, due to the invertibility requirement. In contrast, NODE puts little constraints on the network architecture, but numerical integration can have errors. Handling the initial condition is trivial for NODE while an additional loss term or special architecture is needed for TIPF. As we will show in the experiments, both strategies have merits.

♢ 6.3.3 Formulation

We now describe our algorithm for solving mass-conserving PDEs (6.2). A PDE of this form is determined by $f_t(\cdot;\mu_t) : \mathbb{R}^d \to \mathbb{R}^d$ plus the initial condition μ_0^*. If a probability flow μ_t with flow map Φ_t and velocity field v_t satisfies the following *self-consistency* condition,

$$v_t(x) = f_t(x;\mu_t), \forall x \text{ in the support of } \mu_t, \tag{6.6}$$

then the continuity equation of this flow implies the corresponding PDE (6.2) is solved. Conversely, the velocity field of any solution of (6.2) will satisfy (6.6). Hence, instead of solving (6.2) which is a condition on the density p_t that might be hard to access, we can solve (6.6) which is a more tractable condition. Shen et al. [She+22] and Shen and Wang [SW24] develop this concept for the Fokker-Planck equation and McKean-Vlasov type PDEs; here we generalize it to a wider class of PDEs of the form (6.2).

Let θ be the network weights that parameterize the probability flow using TIPF or NODE. The flow's measure, velocity field, and flow map at time t are denoted as μ_t^θ, v_t^θ, Φ_t^θ respectively. One option to solve (6.6) would be to minimize the *self-consistency loss*

$$\min_\theta \int_0^T \mathbb{E}_{X\sim\mu_t^\theta}\left[\|v_t^\theta(X) - f_t(X;\mu_t^\theta)\|^2\right] \mathrm{d}t. \tag{6.7}$$

This formulation is reminiscent of PINNs [RPK19] where a residual of the original PDE is minimized. Direct optimization of (6.7) is challenging: while the integration over $[0,T]$ and μ_t^θ can be approximated using Monte Carlo, to apply stochastic gradient descent, we must differentiate through μ_t^θ and f_t: this can be either expensive or intractable depending on the network parameterization. The algorithm by Shen and Wang [SW24] minimizes (6.7) with the adjoint method specialized to Fokker-Planck equations and McKean-Vlasov type PDEs; extending their approach to more general PDEs requires a closed-form formula for the time evolution of the quantities within f_t, which at best can only be obtained on a case-by-case basis.

Instead, we propose the following iterative optimization algorithm to solve (6.7). Let θ_k denote the network weights at iteration k. We define iterates

$$\theta_{k+1} \triangleq \theta_k - \eta\nabla_\theta|_{\theta=\theta_k} F(\theta,\theta_k), \tag{6.8}$$

where

$$F(\theta,\theta_k) \triangleq \int_0^T \mathbb{E}_{X\sim\mu_t^{\theta_k}}\left[\|v_t^\theta(X) - f_t(X;\mu_t^{\theta_k})\|^2\right] \mathrm{d}t. \tag{6.9}$$

Effectively, in each iteration, we minimize (6.9) by one gradient step where we match the velocity field v_t^θ to what it should be according to f_t based on the network weights θ_k from the previous iteration. This scheme can be interpreted as a gradient descent on 6.7 using the biased gradient estimate $\nabla_\theta F(\theta,\theta_k)$—see Sec. 6.A for a discussion. We call this iterative algorithm *self-consistent velocity matching*.

If f_t depends on the density of μ_t only through the score $\nabla\log p_t$ (corresponding to a diffusion term in the PDE), then we can apply an integration-by-parts trick [HD05] to get rid of this density dependency by adding a divergence term of the velocity field. Suppose f_t is from the Fokker-Planck equation (6.4). Then the cross term in (6.9) after expanding the squared norm has the following alternative expression.

Proposition 6.3.1. *For every $t\in[0,T]$, for f_t defined in (6.4), assume v_t^θ, D_t are bounded and continuously differentiable, and $\mu_t^{\theta'}$ is a measure with a continuously differentiable density $p_t^{\theta'}$ that vanishes in infinity and not at finite points. Then we have*

$$\mathbb{E}_{X\sim\mu_t^{\theta'}}\left[v_t^\theta(X)^\top f_t(X;\mu_t^{\theta'})\right] = \mathbb{E}_{X\sim\mu_t^{\theta'}}\left[v_t^\theta(X)^\top b_t(X) + \nabla\cdot\left(D_t^\top(X)v_t^\theta(X)\right)\right]. \tag{6.10}$$

We provide the derivation in Sec. 6.B. With Prop. 6.3.1, we no longer need access to $\nabla \log p_t$ when computing $\nabla_\theta F$. This is useful for NODE parameterization since obtaining the score would otherwise require additional numerical integration.

Our algorithm is summarized in Alg. 6.3.1. We use Adam optimizer [KB14] to modulate the update (6.8). For sampling time steps $t_1, \dots, t_L$ in $[0, T]$, we use stratified sampling where t_l is uniformly sampled from $[\frac{(l-1)T}{L}, \frac{lT}{L}]$; such a sampling strategy results in more stable training in our experiments. We implemented our method using JAX [Bra+18] and FLAX [Hee+20]. See Sec. 6.C for the implementation details.

Algorithm 6.3.1 Self-consistent velocity matching

Input: $f_t(\cdot, \cdot)$, μ_0^*, T, N_{train}, B, L
Initialize network weights θ
for $k = 1, \dots, N_{\text{train}}$ **do**
  $\theta' \leftarrow \theta$
  Sample $x_1, \dots, x_B \sim \mu_0^*$, $t_1, \dots, t_L \sim [0, T]$
  $y_{b,l} \leftarrow \Phi_{t_l}^{\theta'}(x_b)$, $\forall b = 1, \dots, B, l = 1, \dots, L$
  $\theta \leftarrow \theta - \eta \nabla_\theta \frac{1}{BL} \sum_{b,l} \| v_{t_l}^\theta(y_{b,l}) - f_{t_l}(y_{b,l}; \mu_{t_l}^{\theta'}) \|^2$
end for
Output: optimized θ

■ 6.4 Experiments

We show the efficiency and accuracy of our method on several PDEs of the form (6.2). We start with three Wasserstein gradient flow experiments (Sec. 6.4.1, Sec. 6.4.2, Sec. 6.4.3). Next, we consider the time-dependent Fokker-Planck equation that simulates attraction towards a harmonic mean in Sec. 6.4.4. Finally, in Sec. 6.4.5, we apply our framework to generate complicated low-dimensional dynamics including flows splashing against obstacles and smooth interpolation of measures. We will use SCVM-TIPF and SCVM-NODE to denote our method with TIPF and NODE parameterization respectively. We use JKO-ICNN to denote the method by Mokrov et al. [Mok+21], JKO-ICNN-PD to denote the method by Fan, Taghvaei, and Chen [FTC21] (PD for "primal-dual"), ADJ to denote the adjoint method by Shen and Wang [SW24], SDE-EM to denote the Euler-Maruyama method for solving the SDE associated with the Fokker-Planck equation, and DFE ("discrete forward Euler") to denote the method by Boffi and Vanden-Eijnden [BV23]. We implemented all competing methods in JAX—see more details in Sec. 6.C—and we compare quantitatively against these methods when possible.

In Tab. 6..1, we compare the time complexity of training the described methods, where we show that SCVM-TIPF and SCVM-NODE have low computational complexity among

all methods.

Evaluation metrics. For quantitative evaluation, we use the following metrics. To compare measures with density access, following Mokrov et al. [Mok+21], we use the symmetric Kullback-Leibler (symmetric KL) divergence, defined as $\mathrm{SymKL}(\rho_1, \rho_2) \triangleq \mathrm{KL}(\rho_1 \parallel \rho_2) + \mathrm{KL}(\rho_2 \parallel \rho_1)$, where $\mathrm{KL}(\rho_1 \parallel \rho_2) \triangleq \mathbb{E}_{X\sim\rho_1}[\log \frac{\mathrm{d}\rho_1(X)}{\mathrm{d}\rho_2(X)}]$. Sample estimates of KL can be negative which complicates log-scale plotting, so when this happens, we consider an alternative f-divergence $D_f(\rho_1 \parallel \rho_2) \triangleq \mathbb{E}_{X\sim\rho_2}[\frac{(\log\rho_1(X)-\log\rho_2(X))^2}{2}]$ whose sample estimates are always non-negative. We similarly define the symmetric f-divergence $\mathrm{Sym}D_f(\rho_1, \rho_2) \triangleq D_f(\rho_1 \parallel \rho_2) + D_f(\rho_2 \parallel \rho_1)$. For particle-based methods, we use kernel density estimation (with Scott's rule) to obtain the density function before computing symmetric KL or f-divergence. We also consider the Wasserstein-2 distance [Bon+11] and the Bures-Wasserstein distance [KSS21]; these two measures only require sample access. All metrics are computed using i.i.d. samples. See Sec. 6.C.6 for more details.

♢ 6.4.1 Sampling from mixtures of Gaussians

We consider computing the Wasserstein gradient flow of the KL divergence $\mathcal{F}(\mu) = \mathrm{KL}(\mu \parallel \mu^*)$ where we have density access to the target measure μ^*. To fit into our framework, we set $f_t(x; \mu_t) = \nabla \log p^*(x) - \nabla \log p_t(x)$ which matches (6.4) with $b_t(x) = \nabla \log p^*(x)$ and $D_t(x) = I_d$. Following the experimental setup in Mokrov et al. [Mok+21] and Fan, Taghvaei, and Chen [FTC21], we take μ^* to be a mixture of 10 Gaussians with identity covariance and means sampled uniformed in $[-5, 5]^d$. The initial measure is $\mu_0^* = \mathcal{N}(0, 16I_d)$. We solve the corresponding Fokker-Planck PDE for a total time of $T = 5$ and for $d = 10, \ldots, 60$. As TIPF parameterization does not scale to high dimensions, we only consider SCVM-NODE in this experiment.

Fig. 1 shows the probability flow produced by SCVM-NODE in dimension 60 at different time steps; as we can see, the flow quickly converges to the target distribution.

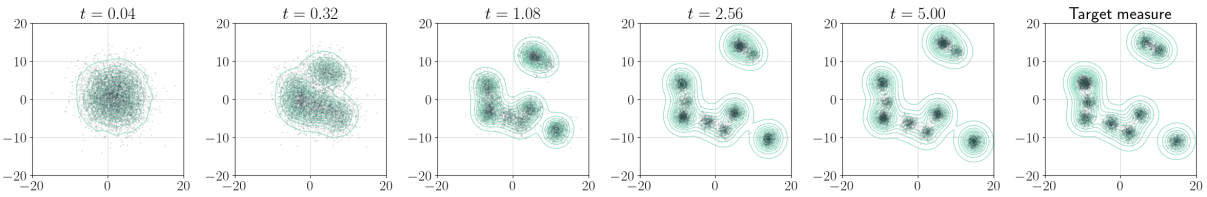

Figure 1: Probability flow produced by SCVM-NODE for a 60-dimensional mixture of Gaussians at irregular time steps. Samples are projected onto the first two PCA components and kernel density estimation is used to generate the contours.

In Fig. 2, we quantitatively compare our method with Mokrov et al. [Mok+21], Fan,

Taghvaei, and Chen [FTC21], and Shen and Wang [SW24]. Training time is reported for all methods.

- In the left two columns of Fig. 2, we find that even though the adjoint method ADJ [SW24] minimizes the self-consistency loss (6.7) directly, the decay of self-consistency can be much slower than that of SCVM-NODE as the dimension increases. We suspect this is due to the amount of error accumulated in the adjoint method which involves two numerical integration passes to obtain the gradient. Moreover, ADJ requires up to *third-order spatial derivatives* of the parameterized neural velocity field which can be inaccurate even if the consistency loss is low—in comparison SCVM-NODE only requires one integration pass and the first-order spatial derivative of the network. Despite the bias of the gradient used in SCVM-NODE, it finds more efficient gradient trajectories than ADJ. Additionally, ADJ takes 80 times longer to train than SCVM-NODE in dimension 10, and scaling up to higher dimensions becomes prohibitive.

- The rightmost column of Fig. 2 shows SCVM-NODE achieves far lower symmetric KL compared to the JKO methods. The gradient flow computed by JKO methods does not decrease KL divergence monotonically, likely because the optimization at each JKO step has yet to reach the minimum even though we use 2000 gradient updates for each step. For both JKO methods, the running time for each JKO step increases linearly because samples (and for JKO-ICNN also log det terms) need to be pushed through a growing chain of ICNNs; as a result, the total running time scales *quadratically* with the number of JKO steps. JKO methods also take about 40 times as long evaluation time as SCVM-NODE in dimension 60 since density access requires solving an optimization problem for every JKO step. On top of the computational advantage and better results, our method also does not have temporal discretization: after being trained, the flow can be accessed at any time t (Fig. 1).

♢ 6.4.2 Ornstein-Uhlenbeck process

We consider the Wasserstein gradient flow of the KL divergence with respect to a Gaussian with the initial distribution being a Gaussian, following the same experimental setup in Mokrov et al. [Mok+21] and Fan, Taghvaei, and Chen [FTC21]. In this case, the gradient flow at time t is a Gaussian $G(t)$ with a known mean and covariance; see Sec. 6.D.1 for details. We quantitatively compare all methods in Fig. 3:

- ADJ achieves the best results in dimensions $d = 5$ and $d = 10$. However, this is at the cost of high training time: in dimension 10, ADJ takes 341 minutes to train, while SCVM-TIPF and SCVM-NODE take 23 and 9 minutes respectively for the same 20k training iterations. As such, we omit ADJ in higher-dimensional comparisons.

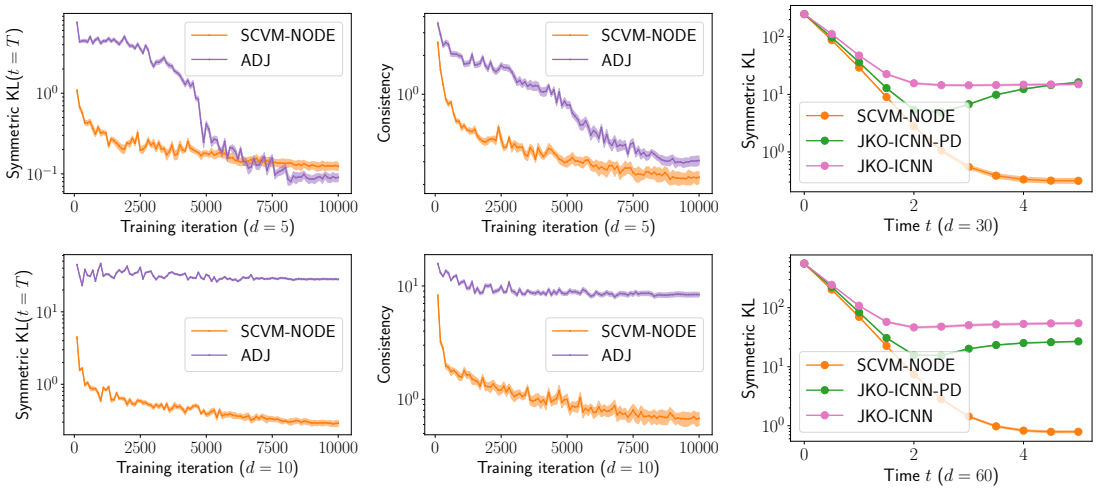

Figure 2: Quantitative comparison for the mixture of Gaussians experiment. The left two columns plot the symmetric KL (at $t = T$ compared against the target measure) and consistency (6.7) versus the training iterations for SCVM-NODE (ours) and ADJ [SW24]. The rightmost column plots the symmetric KL across time t (compared against the target measure) for SCVM-NODE and the JKO methods in high dimensions. Training time: for $d = 10$, SCVM-NODE takes 7.37 minutes, ADJ takes 585.2 minutes; for $d = 60$, SCVM-NODE takes 23.9 minutes, JKO-ICNN takes 375.2 minutes, and JKO-ICNN-PD takes 24.4 minutes.

- Both our SCVM-TIPF and SCVM-NODE achieve good results second only to ADJ in dimensions 5 and 10. In low dimensions, SCVM-TIPF results in lower probability divergences than SCVM-NODE likely due to having exact density access. Although not shown, SCVM-TIPF also satisfies the initial condition well (numbers at $t = 0$ are comparable to those at $t = 0.25$ in the left two columns in Figure 3).
- For the two JKO methods, they result in much higher errors for $t \leq 0.5$ compared to later times: this is expected because the dependency of $G(t)$ on t is exponential, so convergence to μ^* is faster in the beginning, yet a constant step size is used for JKO methods.
- For DFE, the result is highly sensitive to the forward Euler step size Δt. We choose step size $\Delta t = 0.01$ which empirically gives the best results among $\{0.1, 0.01, 0.001, 0.0001\}$. As DFE achieves far lower symmetric KL divergence or f-divergence compared to alternatives, we only include its Bures-Wasserstein distance in high dimensions where its number is slightly worse than alternatives (bottom right plot of Fig. 3).

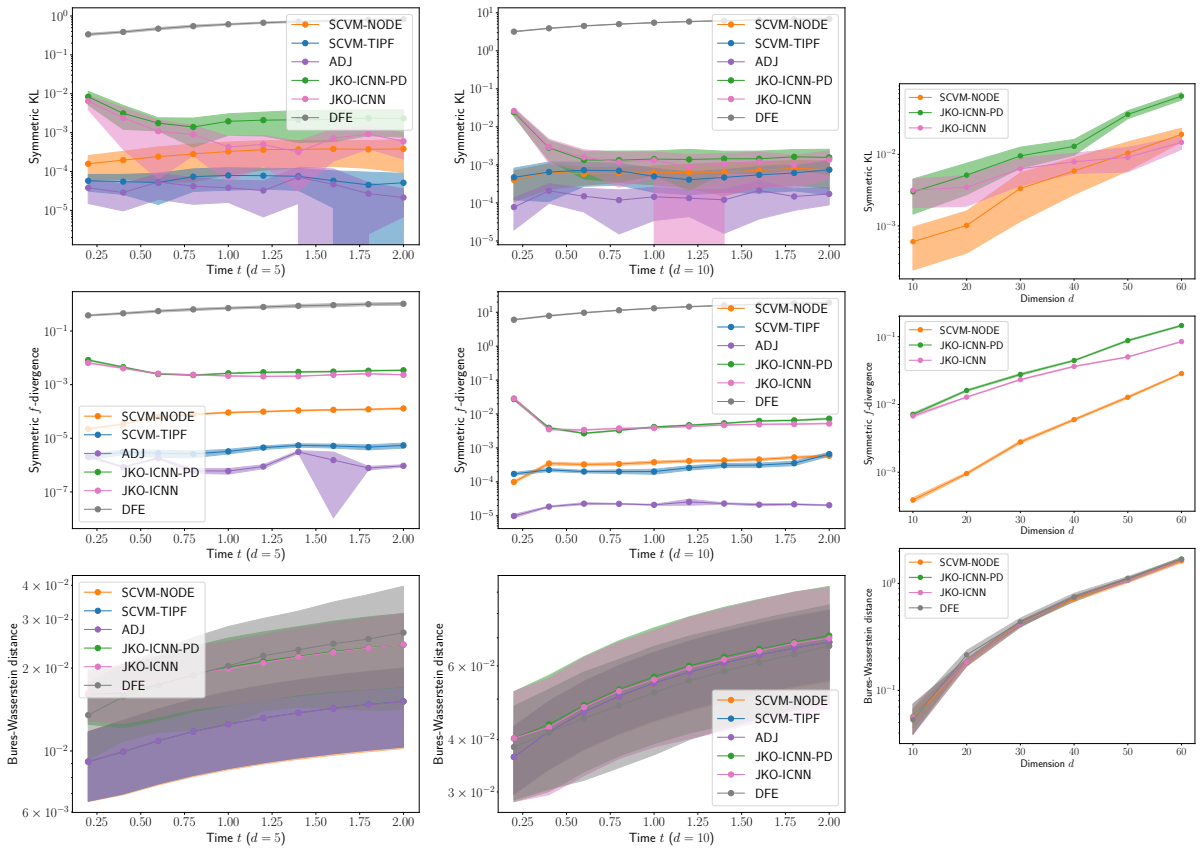

Figure 3: Quantitative results for the OU process experiment. The left two columns show the metrics (symmetric KL, symmetric f-divergence, and Bures-Wasserstein distance) versus time t of various methods computed against the closed formed solution $G(t)$ in dimension $d = 5, 10$. The right column shows the metrics averaged across t versus dimension d in higher dimensions.

♢ 6.4.3 Porous medium equation

Following Fan, Taghvaei, and Chen [FTC21], we consider the porous medium equation with only diffusion: $\partial_t p_t = \Delta p_t^m$ with $m > 1$. Its solution is the Wasserstein gradient flow of $\mathcal{F}(\mu) = \int \frac{1}{m-1} p(x)^m \mathrm{d}x$ where p is the density of μ with $\nabla_{W_2}\mathcal{F}(\mu)(x) = \nabla(\frac{m}{m-1}p^{m-1}(x))$—see Sec. 6.D.2 for details. We consider only SCVM-TIPF and JKO methods here.

We show the efficiency of SCVM-TIPF compared to JKO-ICNN in dimension $d = 1, 2, \ldots, 6$. We exclude JKO-ICNN-PD because it produces significantly worse results. We visualize the density p_t of the solution from SCVM-TIPF and JKO-ICNN on the top of Fig. 4 in dimension 1 compared to p_t^*. Both methods approximate p_t^* well with SCVM-

TIPF being more precise at small t; this is consistent with the observation in Fig. 2 where JKO methods result in bigger errors for small t.

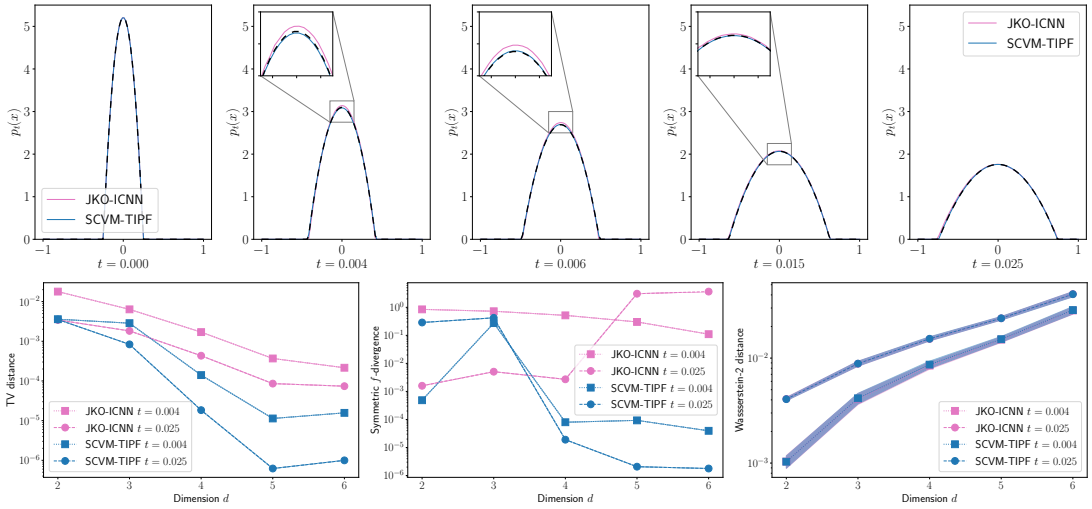

Figure 4: Top: visualization of the densities of p_t^* and p_t for the porous medium equation in dimension 1 at varying time steps t for SCVM-TIPF and JKO-ICNN. Bottom: total variation distance, symmetric f-divergence, and Wasserstein-2 distances across dimensions at $t = 0.004$ and $t = 0.025$ between p_t and p_t^* for solving the porous medium equation.

On the bottom row of Fig. 4, we plot the total variation (TV) distance, the symmetric f-divergence, and the Wasserstein-2 distance (details on the TV distance are given in Appendix 6.C.6) between the recovered solution p_t and p_t^* for both methods at $t = 0.004$ and $t = 0.025$. Note that the values of all metrics are very low implying that the solution from either method is very accurate, with SCVM-TIPF more precise in TV distance and symmetric f-divergence, especially for $d > 3$. Like with the experiments in previous sections, JKO-ICNN is much slower to train: in dimension 6, training JKO-ICNN took 102 minutes compared to 21 minutes for SCVM-TIPF.

♢ 6.4.4 Time-dependent Fokker-Planck equation

In this section, we qualitatively evaluate our method for solving a PDE that is not a Wasserstein gradient flow. In this case, JKO-based methods cannot be applied. Consider the OU process from Sec. 6.4.2 when the mean β and the covariance matrix Γ become time-dependent as β_t and Γ_t. The resulting PDE is a time-dependent Fokker-Planck equation of the form (6.4) with

$$f_t(X, \mu_t) = \Gamma_t(\beta_t - X) - D\nabla \log p_t(X). \tag{6.11}$$

In this configuration, when the initial measure p_0 is Gaussian, the solution μ_t can again be shown to be Gaussian with mean and covariance following an ODE—see Sec. 6.D.3 for more details. We consider, in dimension 2 and 3, time-dependent attraction towards a harmonic mean $\beta_t = a(\sin(\pi\omega t), \cos(\pi\omega t))$ using the expression of β_t from Boffi and Vanden-Eijnden [BV23], augmented to $\beta_t = a(\sin(\pi\omega t), \cos(\pi\omega t), t)$ in dimension 3.

We apply both SCVM-TIPF and SCVM-NODE to this problem and compare our results with alternatives. Similar to Fig. 3, as shown in Fig. 5, both SCVM-TIPF and SCVM-NODE achieve results on par with ADJ, with both SCVM methods being 30 times faster than ADJ in dimension 10. DFE results in good Wasserstein-2 metrics but worse divergences. Visualization of the evolution of a few sampled particles are given in Fig. 6.D.2 and Fig. 6.D.3.

In Sec. 6.D.4, we augment (6.11) with an interaction term to simulate a flock of (infinitely many) birds, resulting in a non-Fokker-Planck PDE that can be readily solved by our method.

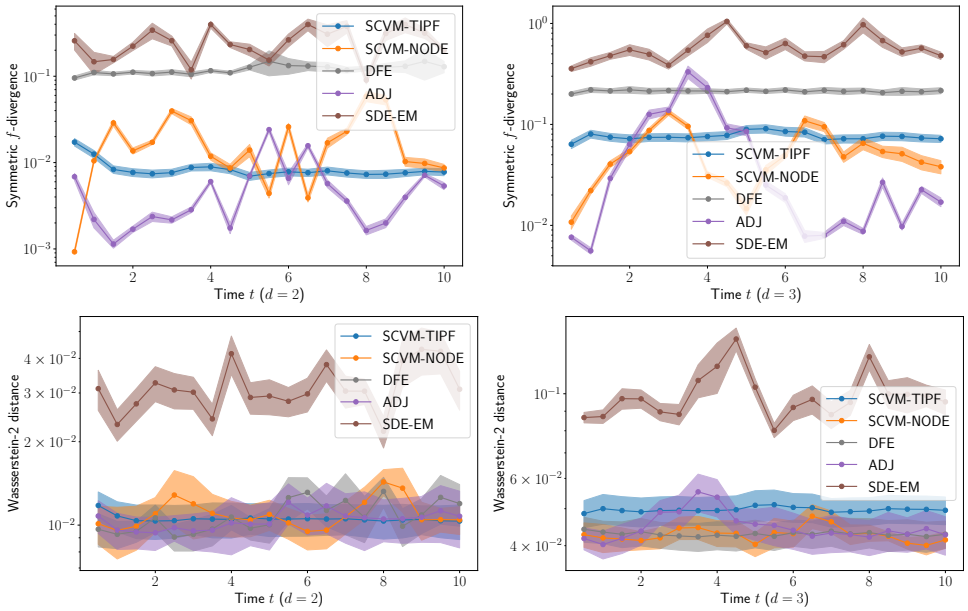

Figure 5: Symmetric KL divergence and Wasserstein-2 distances across time for $d = 2, 3$ between the recovered flows and the ground truth for the time-dependent Fokker-Planck equation.

♢ 6.4.5 Additional qualitative low-dimensional dynamics

To demonstrate the flexibility of our method, we apply our algorithm to model more general mass-conserving dynamics than the ones considered in the previous sections.

Flow splashing against obstacles. We model the phenomenon of a 2-dimensional flow splashing against obstacles using a Fokker-Planck equation (6.4) where b_t encodes the configuration of three obstacles that repel the flow (See Sec. 6.D.5 for details). We solve this PDE using SCVM-NODE for $T = 5$ and visualize the recovered flow in (6). When solving the same PDE using SDE-EM, the flow incorrectly crosses the bottom right obstacle due to a finite time step size (Fig. 6.D.7). When using DFE, the path of initial samples appears jagged (right of Fig. 6.D.6); our method has no such issue and results in continuous sample paths (left of Fig. 6.D.6). Method ADJ suffers from numerical instability and cannot be trained without infinite loss in this example.

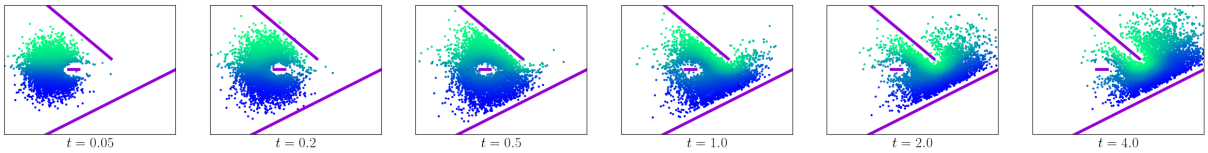

Figure 6: A flow splashing against three obstacles (in purple) produced by SCVM-NODE. Particles are colored based on the initial y coordinates.

Smooth interpolation of measures. To illustrate the flexibility of our method, we demonstrate two ways to formulate the problem of smoothly interpolating a list of measures. First, we model the interpolation as a time-dependent Fokker-Planck equation and use it to interpolate MNIST digits 1, 2, and 3, starting from a Gaussian (Fig. 7). Next, we adopt an optimal transport formulation and use it to generate an animation sequence deforming a 3D hand model to a different pose and then to a ball, similar to the setup in Zhang, Smirnov, and Solomon [ZSS22]. Note that the optimal transport formulation is not solvable using competing methods. See Sec. 6.D.6 for more details.

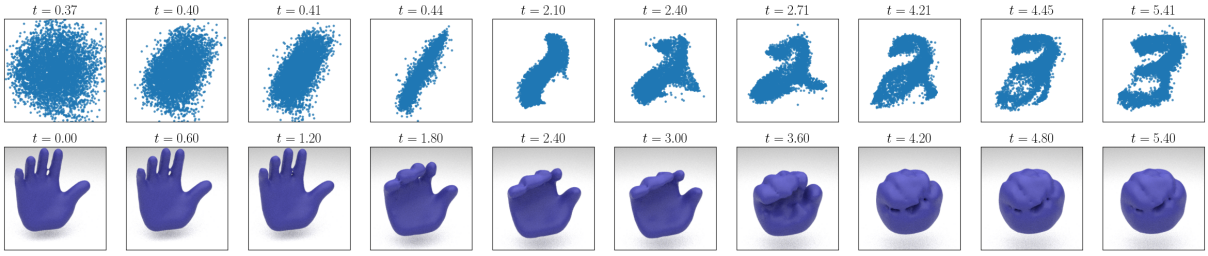

Figure 7: Smooth interpolation of measures. Top: interpolating MNIST digits 1 to 3. Bottom: interpolating hand from the initial pose to a different pose and then to a ball.

6.5 Conclusion

By extending the concept of self-consistency from Shen et al. [She+22], we present an iterative optimization method for solving a wide class of mass-conserving PDEs without temporal or spatial discretization. In all experiments considered, our method achieves strong quantitative results with significantly less training time than JKO-based methods and the adjoint method in high dimensions.

Below we highlight a few future directions. First, as discussed, the two ways to parameterize a probability flow, TIPF, and NODE, both have their specific limitations. Finding a new parameterization that combines the advantages of both TIPF and NODE is an important next step. Secondly, we hope to extend our approach to incorporate more complicated boundary conditions. Finally, given that the proposed algorithm is highly effective empirically, it would be an interesting theoretical step to explore its convergence properties.

Appendices

Appendix

In this appendix, we provide details and further justification of the proposed method. In Sec. 6.A, we provide an interpretation of the update (6.9) as a gradient descent step with a biased gradient. In Sec. 6.B, we explain the integration-by-parts trick used to prove (6.3.1). In Sec. 6.C, we provide implementation details of all considered methods. In Sec. 6.D, we provide additional experimental details and results.

■ 6.A Biased Gradient Interpretation

Assume μ_t^θ has density p_t^θ. Using $f_t^\theta(x)$ to denote $f_t(x;\mu_t^\theta)$ from (6.7), the self-consistency loss can be written as,

$$L(\theta) \triangleq \int_0^T \int p_t^\theta(x)\|v_t^\theta(x) - f_t^\theta(x)\|^2 \mathrm{d}x\mathrm{d}t.$$

SCVM-TIPF	SCVM-NODE	ADJ	JKO-ICNN	JKO-ICNN-PD
$O(STd^2)$	$O(STN_{\text{ode}}d)$	$O(STN_{\text{ode}}d^3)$	$O(ST^2d^3)$	$O(ST^2d)$

Table 6..1: Time complexity of training for S iterations for the methods considered, in terms of dimension d. We assume for simplicity that the same batch size is used, the training is done for T time steps, and any network forward pass takes $O(d)$ time. For SCVM-NODE and ADJ, N_{ode} denotes the number of ODE integration steps. For SCVM-TIPF, RealNVP is used to build the coupling layer, and we use d coupling layers, hence the extra multiple of d. For ADJ, the d^3 term comes from the third-order spatial derivatives. For JKO-ICNN, the d^3 term is due to computing the log-determinant term. For both JKO methods, the quadratic dependency on T is due to maintaining a growing chain of neural networks of size T as described in the related works section.

Assuming all terms involving θ are differentiable with respect to θ, the gradient of $L(\theta)$ with respect to the neural network parameters θ can be written as:

$$\begin{aligned}\nabla L(\theta) = &\int_0^T \int \nabla_\theta p_t^\theta(x) \|v_t^\theta(x) - f_t^\theta(x)\|^2 \mathrm{d}x\mathrm{d}t \\ &+ 2\int_0^T \int p_t^\theta(x) J_\theta v_t^\theta(x)^\top (v_t^\theta(x) - f_t^\theta(x)) \mathrm{d}x\mathrm{d}t \\ &- 2\int_0^T \int p_t^\theta(x) J_\theta f_t^\theta(x)^\top (v_t^\theta(x) - f_t^\theta(x)) \mathrm{d}x\mathrm{d}t.\end{aligned} \tag{6.12}$$

Here we use J_θ to denote the Jacobian with respect to θ. On the other hand, the gradient used in the updates (6.8) is $\nabla_\theta F(\theta, \theta')$ at $\theta' = \theta$:

$$\nabla_\theta F(\theta, \theta')\Big|_{\theta'=\theta} = 2\int_0^T \int p_t^\theta(x) J_\theta v_t^\theta(x)^\top (v_t^\theta(x) - f_t^\theta(x)) \mathrm{d}x\mathrm{d}t. \tag{6.13}$$

We see that (6.13) is exactly the middle term (6.12). Hence our formulation can be interpreted as doing gradient descent with a biased gradient estimator. It remains a future work direction to theoretically analyze the amount of bias in (6.13) and the condition under which the dot product $\langle \nabla L(\theta), \nabla_\theta F(\theta, \theta')|_{\theta'=\theta} \rangle \geq 0$. The central challenge would be to relate $J_\theta v_t^\theta$ and $J_\theta f_t^\theta$; this depends on the neural network architecture and the type of the PDE.

■ 6.B Integration-by-Parts Trick

This is a common trick used in score-matching literature [HD05].

Proof of Prop. 6.3.1. Fix $t > 0$. The form of f_t in (6.4) is

$$f_t(x; \mu_t) = b_t(x) - D_t(x) \nabla \log p_t(x).$$

Hence

$$\mathbb{E}_{X\sim\mu_t^{\theta'}}\left[v_t^\theta(X)^\top f_t(X; \mu_t^{\theta'})\right] = \mathbb{E}_{X\sim\mu_t^{\theta'}}\left[v_t^\theta(X)^\top b_t(X)\right] - \mathbb{E}_{X\sim\mu_t^{\theta'}}\left[v_t^\theta(X)^\top D_t(X) \nabla \log p_t^{\theta'}(X)\right].$$

The second term can be written as

$$\begin{aligned}\mathbb{E}_{X\sim\mu_t^{\theta'}}\left[v_t^\theta(X)^\top D_t(X) \nabla \log p_t^{\theta'}(X)\right] &= \int v_t^\theta(x)^\top D_t(x) \nabla \log p_t^{\theta'}(x) \mathrm{d}p_t^{\theta'}(x) \\ &= \int v_t^\theta(x)^\top D_t(x) \nabla p_t^{\theta'}(x) / p_t^{\theta'}(x) \cdot p_t^{\theta'}(x) \mathrm{d}x \\ &= \int v_t^\theta(x)^\top D_t(x) \nabla p_t^{\theta'}(x) \mathrm{d}x \\ &= -\int \nabla \cdot (D_t(x)^\top v_t^\theta(x)) p_t^{\theta'}(x) \mathrm{d}x \\ &= -\mathbb{E}_{X\sim\mu_t^{\theta'}}\left[\nabla \cdot (D_t(X)^\top v_t^\theta(X))\right],\end{aligned}$$

where we use integration-by-parts to get the second last equation and the assumption that v_t^θ, D_t are bounded and $p_t^\theta(x) \to 0$ as $\|x\| \to \infty$. □

■ 6.C Implementation Details

♢ 6.C.1 Network architectures for SCVM

For TIPF, our implementation follows Dinh, Sohl-Dickstein, and Bengio [DSB16]. Each coupled layer uses 3-layer fully connected networks with layer sizes 64, 128, 128 for both scale and translation prediction. We use twice as many coupling layers as the dimension of the problem while each coupling layer updates one coordinate; we found using fewer layers with random masking gives much worse results.

For NODE, we use a 3-layer fully connected network for modeling the velocity field with layer size 256. We additionally add a low-rank linear skip connection $x \mapsto A(t)x+b(t)$ where $A(t) = L(t)L^\top(t)$ and $L(t)$ is a $d \times 20$ matrix to make $A(t)$ low-rank.

We use SILU activation [EUD18] which is smooth for all layers for both TIPF and NODE. For NODE, we apply layer normalization before applying activation. We also add a sinusoidal embedding for the time input t plus two fully connected layers of size 64 before concatenating it with the spatial input.

The numerical integration for NODE is done using Diffrax library [Kid21] with a relative and absolute tolerance of 10^{-4}; we did not find considerable improvement when using a lower tolerance.

We use the integration-by-parts trick for SCVM-NODE whenever possible. Since TIPF has tractable log density, we do not use such a trick and optimize (6.9) directly for SCVM-TIPF which we found to produce better results.

♢ 6.C.2 Hyperparameters

Unless mentioned otherwise, we choose the following hyperparameters for Alg. 6.3.1. We set $N_{\text{train}} = 10^5$ or 2×10^5, $B = 1000$, $L = 10$ or 20. We use Adam [KB14] with a cosine decay learning rate scheduler, with initial learning rate 10^{-3}, the number of decay steps same as N_{train}, and $\alpha = 0.01$ (so the final learning rate is 10^{-5}). Since we are effectively performing gradient descent using a biased gradient, we set $b_2 = 0.9$ in Adam (instead of the default $b_2 = 0.999$), so that the statistics in Adam can be updated more quickly; we found this tweak improves the results noticeably.

♢ 6.C.3 Implementation of JKO methods

We base our JAX implementation of ICNN on the codebase by the original ICNN author: `https://github.com/facebookresearch/w2ot`. Compared to the original ICNN imple-

mentation by Amos, Xu, and Kolter [AXK17], we add an additional convex quadratic skip connections used by Mokrov et al. [Mok+21], which we found to be crucial for the OU process experiment. For ICNNs, we use hidden layer sizes 64, 128, 128, 64. The quadratic rank for the convex quadratic skip connections is set to 20. The activation layer is taken to be CELU.

To implement the method by Fan, Taghvaei, and Chen [FTC21], we model the dual potential as a 4-layer fully connected network with layer size 128, with CELU activation. For the gradient flow of KL divergence and generalized entropy (used in Sec. 6.4.3), we follow closely the variational formulation and the necessary change of variables detailed in Fan, Taghvaei, and Chen [FTC21, Corollary 3.3, Corollary 3.4].

In order to compute the log density at any JKO step, following Mokrov et al. [Mok+21], we need to solve a convex optimization to find the inverse of the gradient of an ICNN. We use the LBFGS algorithm from JAXopt [Blo+22] to solve the optimization with tolerance 10^{-2} (except for Sec. 6.4.3 we use a tolerance of 10^{-3} to obtain finer inverses, but it takes 6x longer compared to 10^{-2}).

We always use 40 JKO steps, consistent with past works. For each JKO step, we perform 1000 stochastic gradient descent using Adam optimizer with a learning rate of 10^{-3}, except for the mixture of Gaussians experiment, we use 2000 steps—using fewer steps will result in worse results. We have tested with the learning rate schedules used in Fan, Taghvaei, and Chen [FTC21] and Mokrov et al. [Mok+21] and did not notice any improvement.

♢ 6.C.4 Implementation of ADJ method

We implement the adjoint method carefully following the formulation in Shen and Wang [SW24]. The neural network for parameterizing the velocity field is identical to the one used in SCVM-NODE. The ODE integration also uses the same hyperparameters as that of SCVM-NODE. This way we can compare ADJ with SCVM-NODE in a fair manner since they only differ in the gradient estimation.

♢ 6.C.5 Implementation of DFE method

We implement DFE following the algorithm outlined in Boffi and Vanden-Eijnden [BV23]. We use 5000 particles. For score estimation, we use the same network architecture as in NODE. At each time step, we optimize the score network 100 steps. We found the result of DFE depends tremendously on the time step size Δt. For the OU process experiment in dimension 60, when $\Delta t = 0.1, 0.01, 0.001, 0.0001$, the resulting Bures-Wasserstein distance at the final time to the target measure is $28.11, 0.31, 0.46, 9.21$ respectively. Surprisingly, a smaller Δt can result in bigger errors. We choose $\Delta t = 0.01$ since it gives the best results.

♢ 6.C.6 Evaluation metrics

For all our experiments, calculations of all metrics are repeated 20 times on 1000 samples from each distribution. Our plots show both the average and the standard deviation calculated over these 20 repetitions.

When estimating symmetric KL divergence using samples, due to the finite sample size and the numerical error in estimating the log density, the estimated divergence can be very close to zero or even negative (when this occurs we take absolute values). This explains why the standard deviation regions touch the x-axis in the log-scale plots in Fig. 3.

To compute Bures-Wasserstein distance [KSS21], we first fit a Gaussian to the samples of either distribution and then compute the closed-form Wasserstein-2 distance between the two Gaussians.

For the porous medium equation (Sec. 6.4.3), the total variation distance is used in Fig. 4 and Fig. 6.D.1 to compare the estimated and ground-truth solutions. It is approximated by the L_1 distance between the densities calculated over 50000 samples uniformly distributed on the compact $[-1.25x_{\max}, 1.25x_{\max}]$ with $x_{\max} = C/\left(\beta(t+t_0)^{\frac{-2\alpha}{d}}\right)$ being the bound of the support of p_t^*.

■ 6.D Additional Experimental Details

♢ 6.D.1 Ornstein-Uhlenbeck process

The OU process is the Wasserstein gradient flow of the KL divergence with respect to a Gaussian $\mu^* = \mathcal{N}(\beta, \Gamma^{-1})$ where $\beta \in \mathbb{R}^d$ and Γ is a $d \times d$ positive-definite matrix. When the initial distribution is $\mu_0^* = \mathcal{N}(0, I_d)$, the gradient flow at time t is known to be a Gaussian distribution $G(t)$ with mean $(I_d - e^{-t\Gamma})\beta$ and covariance $\Gamma^{-1}(I_d - e^{-2t\Gamma}) + e^{-2t\Gamma}$. We set the total time $T = 2$.

♢ 6.D.2 Porous medium equation

This flow has as closed-form solution given by the Barenblatt profile [Váz07] when initialized accordingly:

$$p_t^*(x) = (t+t_0)^{-\alpha}\left(C - \beta\|x\|^2(t+t_0)^{\frac{-2\alpha}{d}}\right)_+^{\frac{1}{m-1}},$$

where $t_0 > 0$ is the starting time, $\alpha = \frac{m}{d(m-1)+2}$, $\beta = \frac{(m-1)\alpha}{2dm}$, and $C > 0$ is a free constant. Similar to Fan, Taghvaei, and Chen [FTC21], we choose $m = 2$ and total time $T = 0.025$. The initial measure follows a Barenblatt distribution supported in $[-0.25, 0.25]^d$ (C is chosen accordingly) with $t_0 = 10^{-3}$. We use Metropolis-Hastings to sample from μ_0.

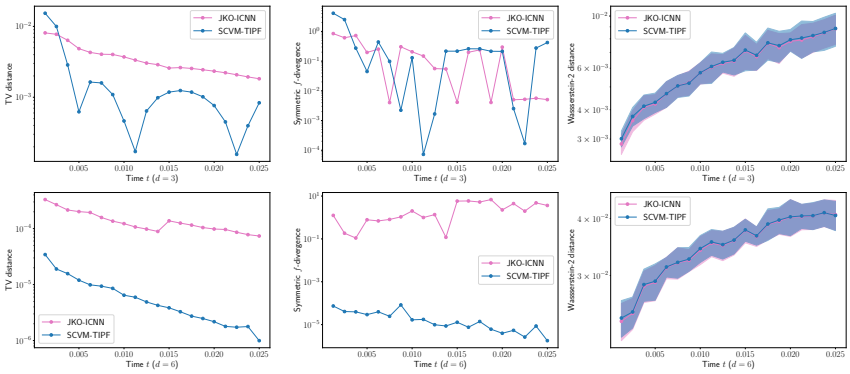

Figure 6.D.1: Metrics (TV, Symmetric f-divergence and Wasserstein-2 distance) across time for dimensions 3 and 6 between the estimated μ_t and the ground-truth μ_t^* when solving the Porous Medium Equation.

♢ 6.D.3 Time-dependant Fokker-Planck equation

We consider a time-dependent Fokker-Planck equation of the form (6.4) with the velocity field

$$f_t(X, \mu_t) = \Gamma_t(X - \beta_t) - D_t \nabla \log p_t(X).$$

When the initial measure p_0 is Gaussian, the solution μ_t can again be shown to be Gaussian with mean m_t and covariance Σ_t solutions of the differential equations:

$$\begin{cases} m_t' &= -\Gamma_t(m_t - \beta_t) \\ \Sigma_t' &= -\Gamma_t \Sigma_t - \Sigma_t \Gamma_t^\top + 2D_t. \end{cases}$$

In practice, we experiment with constant $\Gamma_t = \mathrm{diag}(1, 3)$ and $D_t = \sigma^2 I_d$. We also experience in dimension 3 by considering and $\Gamma_t = \mathrm{diag}(1, 3, 1)$. We set $a = 3$, $\omega = 1$, $\sigma = \sqrt{0.25}$ and pick as initial distribution p_0 a Gaussian with mean b_0 and covariance $\sigma^2 I_d$. We set the total time to $T = 10$.

We plot in Fig. 6.D.2, for dimension 2, snapshots at different time steps of particles following the flow given by our method with TIPF parametrization. We only show SCVM-TIPF because SCVM-NODE gives visually indistinguishable trajectories. We also plot in Fig. 6.D.3 the evolution of particles simulated by Euler-Maruyama (EM-SDE) discretization of the Fokker-Planck equation. Corresponding animated GIFs be found at this link.

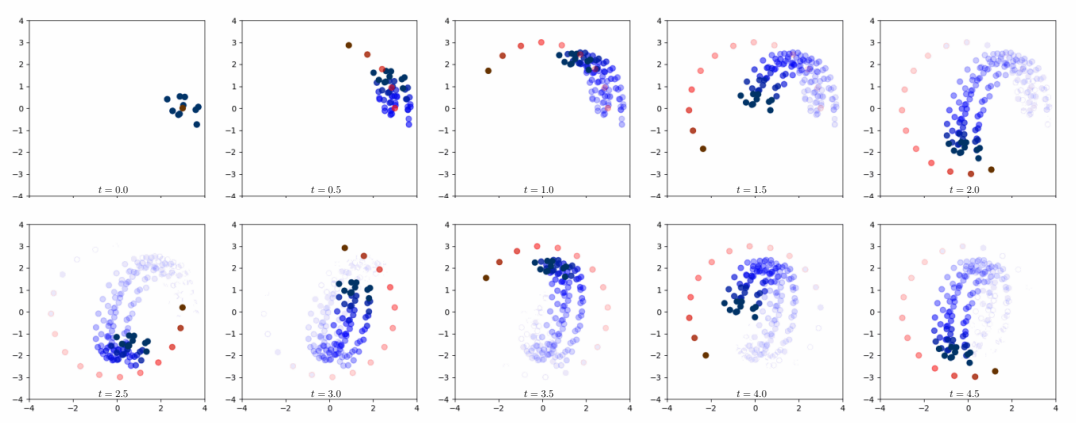

Figure 6.D.2: Evolution of particles (in blue) following the flow learned with SCVM-TIPF for the time-dependent OU process (Sec. 6.4.4). In red is the moving attraction trap.

♢ 6.D.4 Flock of birds

We model the dynamics of a flock of birds by augmenting the time-dependent Fokker-Planck equation (6.11) with an interaction term:

$$f_t(X, \mu_t) = \Gamma_t(\beta_t - X) + \alpha_t(X - \mathbb{E}[\mu_t]) - D\nabla \log p_t(X).$$

This is similar to the harmonically interacting particles experiment in Boffi and Vanden-Eijnden [BV23], but we use a population expectation $\mathbb{E}[\mu_t]$ instead of an empirical one in modeling the repulsion from the mean. Since f_t needs to access $\mathbb{E}[\mu_t]$, the resulting PDE is not a Fokker-Planck equation (6.4) and hence not solvable using the method in Boffi and Vanden-Eijnden [BV23] but can be solved with our method by estimating $\mathbb{E}[\mu_t]$ using Monte Carlo samples from μ_t. We use a similar setup as in Sec. 6.4.4, except we now use an "infinity sign" attraction $\beta_t = a(\cos(2\pi\omega t), 0.5\sin(2\pi\omega t))$ along with a sinusoidal $\alpha_t = 2\sin(\pi w t)$. Depending on the sign of α_t, particles are periodically attracted towards or repulsed from their mean. Both SCVM-TIPF and SCVM-NODE produce similar visual results as shown in Fig. 6.D.4 and Fig. 6.D.5.

We use a constant $\Gamma_t = I_d$ and a constant diffusion matrix $D = \sigma^2 I_d$. We set $a = 3$, $\omega = 0.5$, and $\sigma = \sqrt{0.25}$. We pick as initial distribution p_0 a Gaussian with mean $(0, 0)$ and covariance $\sigma^2 I_d$. We set the total time to $T = 10$.

We respectively show in Fig. 6.D.4 and Fig. 6.D.5 simulations of particles following the flow learned with SCVM-TIPF and SCVM-NODE. Corresponding animated GIFs be found at this link.

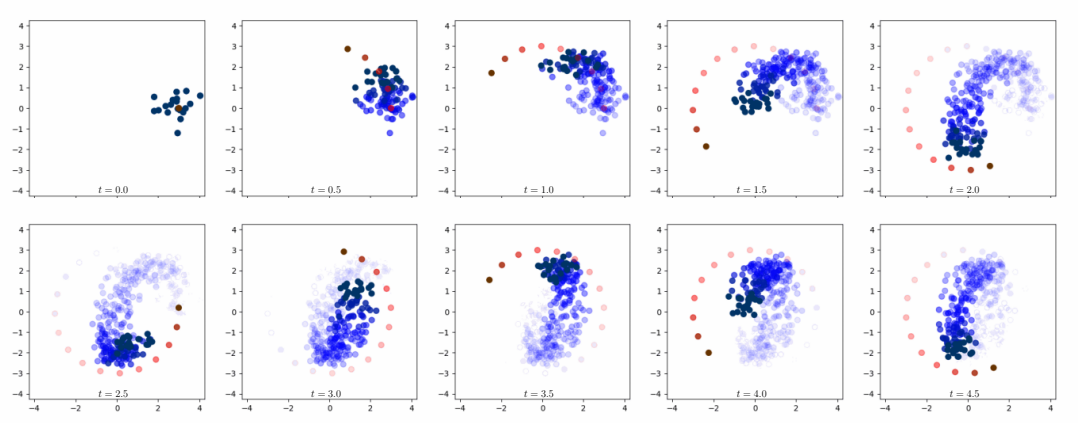

Figure 6.D.3: Evolution of particles (in blue) obtained by SDE-EM discretization for the time-dependent OU process (Sec. 6.4.4). In red is the moving attraction trap.

♢ 6.D.5 Flow splashing against obstacles

We use the following formulation for modeling the flow. Each obstacle is modeled as a line segment. The endpoints of the three obstacles are:

$$((0,3),(3,0.5)),((1,0),(1.5,0)),((-2,-4),(6,0)).$$

We model the dynamics as a Fokker-Planck equation where f_t of the form (6.4) is defined as

$$b_t(x) = (q_{\text{sink}} - x) + 20\sum_{i=1}^{3} \frac{x - \pi_{O_i}(x)}{\|x - \pi_{O_i}(x)\|} p_{\mathcal{N}(0,0.04)}(\|x - \pi_{O_i}(x)\|),$$
$$D_t(x) = I_2,$$

where $q_{\text{sink}} = (4,0)$, and $\pi_{O_i}(x)$ is the projection of x onto obstacle i represented as a line segment, and $p_{\mathcal{N}(0,0.04)}$ is the density of an 1-dimensional Gaussian with variance 0.04.

The initial distribution is chosen to be $\mathcal{N}(0, 0.25I_2)$. We train SCVM-NODE for 10^4 with an initial learning rate of 10^{-4}. Training takes 5.4 minutes. The time step size for SDE-EM used to produce Fig. 6.D.7 is 0.005. Corresponding animated GIFs be found at this link.

♢ 6.D.6 Smooth interpolation of measures

Suppose we are to smoothly interpolate M measures $\nu_1, \ldots, \nu_M$ with densities $q_1, \ldots, q_M$, and we want the flow to approximate ν_i at time r_i. To achieve this goal, we present two formulations using different choices of f_t. We use SCVM-NODE in this section.

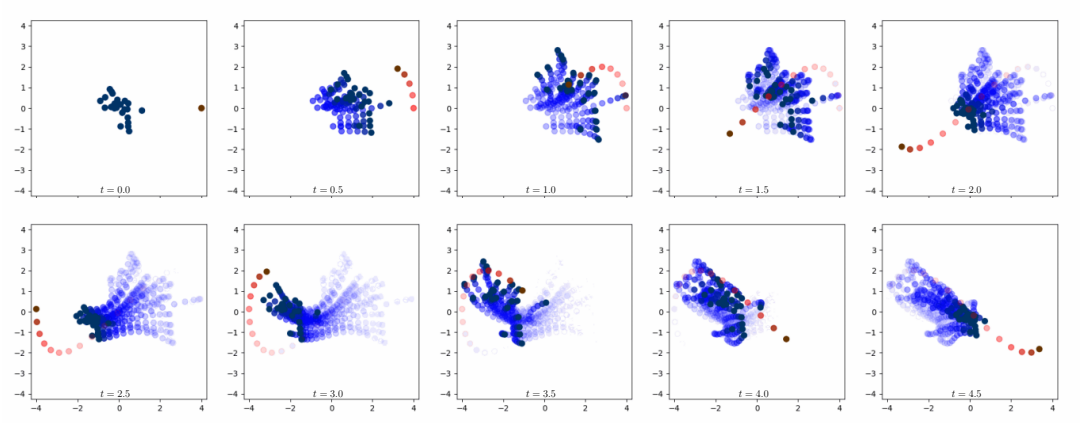

Figure 6.D.4: Evolution of particles following the flow trained with TIPF parametrization on the flock of birds PDE (Section 6.4.5). In red shows the moving attraction mean.

Measure interpolation using time-dependent Fokker-Planck equations. We model the dynamics as a Fokker-Planck equation where f_t of the form (6.4) is taken to be

$$\begin{aligned} b_t(x) &= \sum_{i=1}^{M} \phi(t - r_i)(\nabla \log q_i(x) - \nabla \log p_t(x)) \\ D_t(x) &= I_2, \end{aligned}$$

where $\phi(t)$ is defined as the continuous bump function

$$\phi(t) = \begin{cases} 1.0 & |t| < 0.5h \\ (0.6h - |t|)/(0.1h) & |t| < 0.6h \\ 0.0 & \text{otherwise}, \end{cases}$$

for bandwidth $h = 1.0$.

Below we provide details for the MNIST interpolation result in the top row of Fig. 7. We use the first three images of $1, 2, 3$ from the MNIST dataset. To construct ν_i from a digit image, we use a mixture of Gaussians where we put one equally-weighted Gaussian with covariance $0.02^2 I_2$ on the pixels with values greater than 0.5 (images are first normalized to have values in $[0, 1]$). The initial distribution is $\mathcal{N}((0.5, 0.5), 0.04 I_2)$. To train SCVM-NODE, we use an initial learning rate of 10^{-4} with cosine decay for a total of 10^5 iterations. This takes roughly 1 hour to train.

Measure interpolation using optimal transport. An alternative way to interpolate measures using our framework is to use optimal transport to define f_t. Recall μ_t denotes

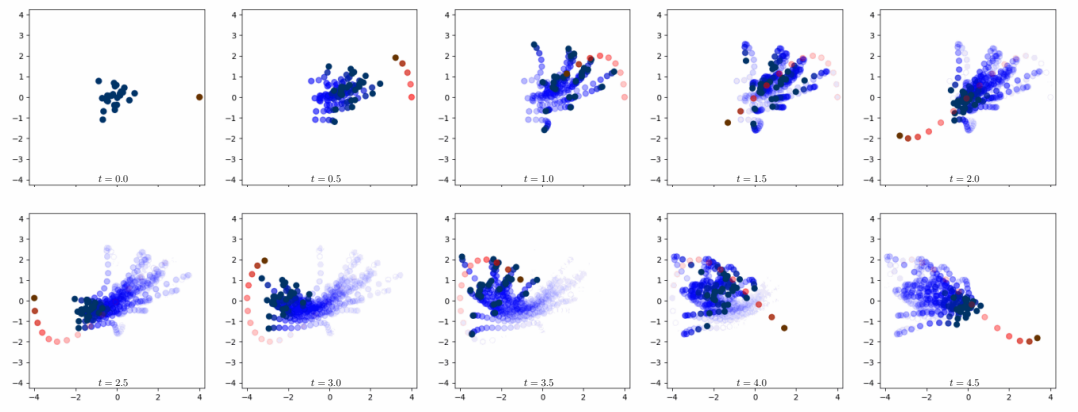

Figure 6.D.5: Evolution of particles following the flow trained with NODE parametrization on the flock of birds PDE (Section 6.4.5). In red shows the moving attraction mean.

the probability flow at time t. We then define

$$f_t(x) = \sum_{i=1}^{M} \phi(t - r_i) \nabla_{W_2} W_2^2(\mu_t, \nu_i),$$

where W_2^2 is the squared Wasserstein-2 distance and $\nabla_{W_2} W_2^2$ is its Wasserstein gradient. In practice, we compute $\nabla_{W_2} W_2^2$ using sample access and we employ the debiased Sinkhorn divergence [GPC18; Fey+19] implemented in the JAX OTT library [Cut+22]. This formulation differs from the one in Zhang, Smirnov, and Solomon [ZSS22] in that here we prescribe the precise PDE based on f_t, whereas in Zhang, Smirnov, and Solomon [ZSS22] an optimal transport loss is used to fit the keyframes along with many regularizers on the velocity field v_t to promote the smoothness and other desirable properties. In contrast, we do not use any regularizer on v_t.

To train SCVM-NODE to produce the hand-hand-ball animation sequence in the bottom row of Fig. 7, we first sample 20000 points from the interior of the three meshes (a hand mesh, a hand mesh in a different pose, and a ball mesh), and we set ν_i to be the empirical measure of the corresponding point cloud. Note that different from the first formulation using Fokker-Planck equations, in the optimal transport formulation, throughout we only require sample access from each ν_i. We use an initial learning rate of 10^{-4} with cosine decay for a total of 5×10^4 iterations. This takes 4.5 hours, which is significantly longer than the training time in Zhang, Smirnov, and Solomon [ZSS22] (reported to be 15 minutes). We leave further improvement of our method to interpolate measures faster as future work.

To render the animation, we sample 20000 points and render the point cloud at each time step using metaballs along with smoothing, similar to the procedure in Zhang,

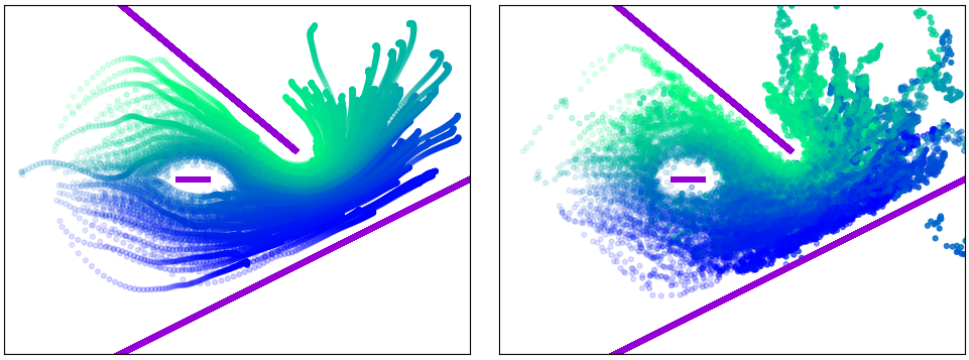

Figure 6.D.6: Trajectory of 200 random particles across time using the same setup as in Fig. 6. Left are sample paths obtained by our method, and right are sample paths obtained by DFE [BV23].

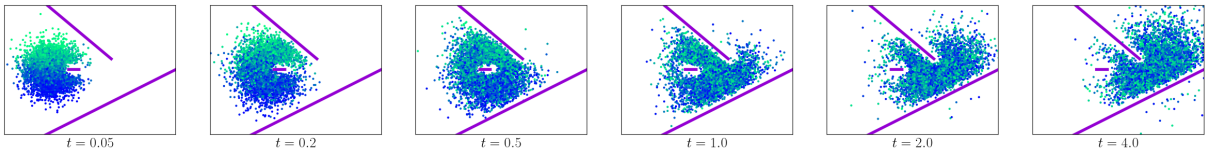

Figure 6.D.7: Same setup as in Fig. 6 but with SDE-EM. We see the paths of the particles are not continuous. Moreover, the particles spill over the obstacle on the bottom right due to a finite time step size. In comparison, SCVM-NODE does not have such a problem.

Smirnov, and Solomon [ZSS22]. We did not use the barycentric interpolation postprocessing step in Zhang, Smirnov, and Solomon [ZSS22] which makes sure the key measures v_i's are fit exactly in the resulting animation. We also did not use unbalanced optimal transport, which as reported in Zhang, Smirnov, and Solomon [ZSS22] can make the fingers of the hand more separated, but requires careful parameter tuning.

Chapter 7

Conclusion

■ 7.1 Common Themes

This thesis has developed scalable methodologies for distributional optimization, showcasing a suite of techniques that address the three challenges—parameterization, modeling, and algorithms—as summarized in Tab. 1.1. These three challenges cannot be addressed in isolation and must be considered holistically in the context of a new distributional optimization problem. While the techniques presented are tailored for specific tasks, they share several common themes that can serve as a guide for future research in this field.

Kernels facilitate optimization over discrete distributions. The key to formulating discrete distributional optmization is to identify the interaction among the discrete population. In Part I, such interaction comes in the form of a kernel—a symmetric, positive definite function. We introduce two methods for generating high-quality samples in Chapters 2 and 3 based on the minimization of an interaction energy of the form

$$\sum_{i,j=1}^{n} w_i w_j \boldsymbol{k}(x_i, x_j),$$

where $(x_i)_{i=1}^n$ are sample points and $(w_i)_{i=1}^n$ are associated weights. Modeling the interaction using a kernel provides several benefits:

(a) The continuous analog of our objective, $\int \boldsymbol{k}(x,y)\mathrm{d}\mu(x)\mathrm{d}\mu(y)$, has a unique minimizer μ^* in the space of distribution under the domain compactness condition (see Prop. 2.1(b)). Moreover, the n-point discrete minimizer converges weakly to μ^* as $n \to \infty$ [BHS19, Cor. 4.2.9]. This discrete-to-continuous connection is essential in deriving the mollified interaction descent algorithm in Chapter 2.

(b) When the sample points are fixed, the problem reduces to a finite-dimensional convex optimization over weights, $\min_w w^\top K w$, where $K \triangleq (\boldsymbol{k}(x_i, x_j))_{ij}$ is positive definite. This structure allows the use of efficient optimization algorithms such as the accelerated mirror descent, as demonstrated in Alg. 3.3.

(c) For a kernel $\boldsymbol{k}$, we can invoke the Mercer representation theorem (Lem. 3.B.1) to obtain the eigenvalues and eigenfunctions of the associated integral operator. We exploited such decomposition to analyze the spectral decay of the kernel (Sec. 3.B.1) and to provide MMD guarantees for the best simplex weights for i.i.d. input (Thm. 3.C.1). Such analysis forms the basis of the debiased compression algorithms introduced in Chapter 3, particularly those based on low-rank-based algorithms, as their guarantees depend on the spectral decay rate of the kernel.

(d) Kernel-based methods, in contrast to Markov chain Monte Carlo approaches, lend themselves well to parallel implementation. We implemented the sampling scheme of Chapter 2 and all subroutines in Chapter 3 using modern machine-learning libraries such as PyTorch and JAX, enabling efficient use of computational resources like GPUs and multi-core CPUs.

Convexity enchances neural network training in continuous optimization. In Part II, we explore how various parameterizations across methods ultimately conform to a unified form:

$$\min_\theta \mathbb{E}_{X\sim\Xi}[F(\Phi_\theta(X), X)], \tag{7.1}$$

where Ξ is a distribution with sample access, $\Phi_\theta : \mathbb{R}^d \to \mathbb{R}^m$ is a neural network with weights θ, and $F : \mathbb{R}^m \times \mathbb{R}^d \to \mathbb{R}$ is a function that is convex and differentiable with respect to the first argument. Specifically,

- In Chapter 4, the objective (4.16) is recast as (7.1) by setting

$$\begin{aligned}
\Xi &= \mu_1 \otimes \cdots \otimes \mu_n \otimes \eta, \\
\Phi_\theta((x_i)_{i=1}^n, y) &= ((f_{\theta_i}(x_i))_{i=1}^n, (g_{\phi_i}(x_i))_{i=1}^n), \\
F((f_i)_{i=1}^n, (g_i)_{i=1}^n, (x_i)_{i=1}^n, y) &= \sum_{i=1}^n \lambda_i \left(R^* \left(f_i + g_i - \sum_{j=1}^n \lambda_j g_j - c(x_i, y) \right) - f_i \right).
\end{aligned}$$

  Here, Φ_θ is a neural network built from smaller neural networks $(f_{\theta_i})_{i=1}^n$ and $(g_{\phi_i})_{i=1}^n$. The function F is convex in the first argument since R^* is convex and the composition of a convex function and an affine function is convex.

- In Chapter 5, to view the objective (5.2) in the form (7.1), we set

$$\Xi = \mu \otimes \nu$$
$$F(\Phi, (x, \tau)) = f_\tau(\Phi) + \frac{\lambda}{2}\|\Phi - x\|_2^2,$$

  which is convex in Φ as long as f_τ is λ-weakly convex.

- In Chapter 6, at iteration $k+1$ of the self-consistent velocity matching algorithm (Alg. 6.3.1), we minimize the objective (6.9) over θ with θ^k being the weights from the previous iteration, which can be viewed in the form (7.1) via

$$\Xi = \text{Uniform}[0, T] \otimes \mu_t^{\theta_k},$$
$$\Phi_\theta(t, x) = v_t^\theta(x),$$
$$F(\Phi, (t, x)) = \|\Phi - f_t(x; \mu_t^{\theta_k})\|_2^2,$$

  where Uniform$[0, T]$ denotes the uniform distribution on $[0, T]$.

In our implementations, we use stochastic gradient descent to minimize (7.1) over network weights where the stochasticity comes from mini-batches sampled from Ξ.

If we minimize (7.1) over the space of all functions $\Phi : \mathbb{R}^d \to \mathbb{R}^m$, then the global optimum Φ^* is obtained *point-wise*, i.e., it satisfies

$$\forall x \in \text{supp}(\Xi), \quad \Phi^*(x) \in \arg\min_y F(y, x). \tag{7.2}$$

Since $F(\cdot, x)$ is convex for each x, intuitively, the gradients will steer the network weights towards the convex set $\arg\min_y F(y, x)$. This intuition is captured in Thm. 5.3.1, where we show that, for a discrete Ξ, training neural networks with an objective like (7.1) converges globally with sufficient overparameterization: the number of network weights is at least $\widetilde{\Omega}(|\text{supp}(\Xi)|)$.

While we have argued that (7.1) can be solved efficiently using neural networks, translating distributional optimization problems into this functional form can be challenging.

Decomposition leads to tractable distributional optimization sub-problems. Throughout, we have demonstrated how complex distributional optimization problems can be effectively decomposed into more manageable sub-problems. For example,

- In Chapter 3, the overall strategy of each debiased compression method is two-phased: first debiasing the input points, followed by compression of the debiased points. Methods that simultaneously debias and compress the input points such as Riabiz et al. [Ria+22] often fail to generate better-than-i.i.d. samples.

- In Chapter 5, while the goal is to find a generative model that generates all minima of a classical optimization problem, we focus on optimizing for the proximal operator as an intermediate step, which enables the proximal-point algorithm during inference to quickly locate multiple minima.
- In Chapter 6, direct minimization of the self-consistency loss (6.7) is challenging due to intractable gradients. Instead, we perform Picard-style fixed-point iterations, where each iteration only requires solving a least-square problem that can be cast in the form of (7.1).

This phenomenon is not limited to the instances studied in this thesis. For example, in diffusion generative modeling [Cao+24], while the end goal is to model a continuous distribution capable of generating fresh samples, the training happens during denoising score matching [Kar+22, Eq. 2] which targets a tractable convex objective of the form (7.1). Subsequently, at inference, the trained denoising model is used as a building block for generation via solving stochastic or deterministic differential equations.

■ 7.2 Future Directions

Our investigation has opened several promising avenues for future work in distributional optimization. Below, we outline key directions suggested by the findings in this thesis.

Breaking the quadratic-time barrier of particle-based sampling methods. While variational inference (VI) methods tend to exhibit faster convergence in practice, the go-to choice for Bayesian inference is still MCMC enabled by many well-tested software packages such as STAN [Car+17]. One central limitation of the particle-based VI methods, such as SVGD [Liu17] and MIED (Chapter 2), is that each update takes $\Omega(n^2)$ time, making it prohibitive to generate more than a few thousand samples. Our investigation suggests two promising directions to break this quadratic-time barrier.

(a) In Chapter 3, we obtain sub-quadratic-time compression algorithms that compress n points to a potentially weighted coreset of size $m \ll n$. In each update step of the particle-based VI methods, instead of aggregating the influence of all n points, we could optionally aggregate the influence only from this coreset, thus reducing the per-iteration time complexity from $\Omega(n^2)$ to $O(nm)$. The challenge is to show the resulting algorithm still converges to the correct distribution with biased update steps. Alternatively, we can apply the debiasing algorithm in Chapter 3 as postprocessing. Furthermore, it is possible to devise a fast incremental coreset update rule to avoid recomputing the coreset from scratch at each iteration.

(b) In Chapter 6, if we were able to evaluate the hypothesis velocity field $f_t(x; \mu_t)$ where μ_t is the empirical measure of the particles, then we could simply move the particles along $f_t(x; \mu_t)$ at each update step. This suggests that we can optimize for a neural network to approximate $f_t(x; \mu_t)$ given sample access to μ_t. Then, we can query the update direction at any given point in constant time by evaluating the network. This idea has been explored in Langosco, Fortuin, and Strathmann [LFS21] and Boffi and Vanden-Eijnden [BV23]. However, several important aspects require deeper investigation. First, how does the approximation error in $f_t(x; \mu_t)$ propagate to an error on the sampling algorithm? Secondly, since the network is trained on a discrete set of points, it can overfit and fail to generalize, yet future evaluations will almost certainly land outside this discrete support; this might explain the high sensitivity of the choice of step sizes for the method by Boffi and Vanden-Eijnden [BV23] as discussed in Sec. 6.4.2. One potential solution is to train with noise added like in denoising diffusion models [Kar+22], but this will likely result in bias.

Capturing mode proportions for multi-modal distributions. As pointed out in Sec. 3.6, when the target distribution has multiple isolated modes, sampling methods that are based solely on the target score, such as SVGD and MIED (Chapter 2), can fail since the score function alone does not capture the relative density between modes. We suggest two ways to incorporate the relative density information.

(a) Bénard, Staber, and Da Veiga [BSD24] suggest using a regularized kernel Stein discrepancy where the regularizer uses density information and leads to correct mode proportions; it is an interesting question whether similar techniques can be employed to improve the debiased compression methods in Chapter 3, and moreover, whether such regularization can be utilized in particle-based VI methods.

(b) Lu, Lu, and Nolen [LLN19] suggest augmenting the overdamped Langevin dynamics with the birth-death process, resulting in a new particle-based sampling method that allows the global move of the probability mass from one mode to another in order to balance the mode proportions. Since the birth-death process requires global information, each step takes time quadratic in the number of samples. It is possible that the ideas mentioned in the previous paragraph could help break this quadratic-time barrier. Another interesting direction is whether we can simulate the same dynamics using the techniques developed in Chapter 6 without having to use particle discretization.

Distributional optimization in higher dimensions. Several of our methods suffer from the curse of dimensionality:

- In Chapter 3, the kernel growth constants in Assum. ($\boldsymbol{\alpha}, \boldsymbol{\beta}$)-kernel often scale linearly with the dimension, yet these constants appear in the exponents of certain terms in the error rates.
- In Chapter 4, prior to optimization, it is necessary to select a proxy for the barycenter measure. We used the uniform measure, which becomes increasingly inadequate in higher dimensions.
- In Chapter 5, the ground space needs to be low-dimensional so that uniform samples, which are used as initial guesses for the proximal-point algorithm, can effectively cover all minima.

While the curse of dimensionality is hard to circumvent in general, we suggest two ideas to mitigate this issue.

(a) While our methods are initially presented with the ground space being Euclidean, most can naturally extend to a lower-dimensional non-Euclidean geometry or a latent space This implies that for addressing high-dimensional distributional optimization problems, one could reduce the dimensionality through representation learning techniques [Zha+18; LHS20] and subsequently atackle the reduced problem in lower dimension. The key challenge is then translating the original problem to a lower-dimensional one, for instance, including how metrics and kernels should be adjusted.

(b) In Chapter 6, our self-consistent velocity matching framework demonstrates significantly better scalability with dimensionality compared to previous state-of-the-art methods for solving mass-conserving dynamics. One reason for this advantage is that in (6.9), the mini-batches used for each gradient update are drawn from the current probability flow, whose interesting dynamics occur in a very small region within the time-space domain. This suggests that more broadly, when optimizing (7.1), employing importance sampling techniques to select mini-batches from a different distribution Ξ' could lead to reduced variance. This will not alter the optimization problem, since the globally optimal solution of (7.1) is obtained point-wise.

Convergence theory involving derivatives of neural networks. An extension of the formulation (7.1) is to allow the objective taking spatial derivatives of the neural network Φ_θ. We have already used such extended formulation in (6.10) thanks to the integration-by-parts trick. Moreover, neural-network-based numerical methods such as Raissi, Perdikaris, and Karniadakis [RPK19] frequently minimize losses that are convex in derivatives of the network output. In these cases, the optimal functional solution is no longer characterized as a point-wise optimal as in (7.2) and will instead exhibit a more complex structure

with non-local dependencies. As a result, the global convergence theory we applied in Chapter 5 is no longer applicable. Interestingly, in practice, we still observe the loss decreasing to zero (e.g. Fig. 2). On the other hand, when the space is discrete, opimization the variable becomes a vector, and the continuous differential operators can be replaced by matrices without altering the convexity of the problem at hand (e.g. Sorkine et al. [Sor+04]). This hints that the same global convergence result might still apply, yet a formal convergence theory studying this aspect is missing.

Convergence of neural networks training with infinite data. Each method described in Part II solves an optimization problem of the form (7.1) by performing stochastic gradient descent using mini-batches of samples from a distribution Ξ with a continuous support. Aside from the scenarios considered in this thesis, the need to compute expectations over a continuous measure with infinite data arises frequently, in particular in low-dimensional settings such as in Raissi, Perdikaris, and Karniadakis [RPK19]. On the other hand, existing convergence theories of overparameterized neural networks [KH19; ALS19] typically assume training on a finite dataset, and the number of network parameters required to obtain global training convergence increases as the dataset size grows. Understanding the convergence and generalization of deep neural networks in the infinite-data regime using stochastic gradient descent remains an unresolved yet critical question. Empirically, we have found using infinite data mitigates overfitting issues. On a different note, so far we have been using i.i.d. samples to form mini-batches, and it is worth exploring whether using better-than-i.i.d. samples, e.g., using the techniques from Chapter 3, would lead to faster convergence.

References

[Aba+16] Martin Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, and Michael Isard. "Tensorflow: A system for large-scale machine learning". In: *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*. 2016, pp. 265–283.

[AC11] Martial Agueh and Guillaume Carlier. "Barycenters in the Wasserstein space". In: *SIAM Journal on Mathematical Analysis* 43.2 (2011), pp. 904–924.

[AC21] Kwangjun Ahn and Sinho Chewi. "Efficient constrained sampling via the mirror-Langevin algorithm". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 28405–28418.

[AGS05] Luigi Ambrosio, Nicola Gigli, and Giuseppe Savaré. *Gradient flows: in metric spaces and in the space of probability measures*. Springer Science & Business Media, 2005.

[AKW12] Sungjin Ahn, Anoop Korattikara, and Max Welling. "Bayesian posterior sampling via stochastic gradient Fisher scoring". In: *arXiv preprint arXiv:1206.6380* (2012).

[ALS19] Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. "A convergence theory for deep learning via over-parameterization". In: *International Conference on Machine Learning*. PMLR. 2019, pp. 242–252.

[Álv+16] Pedro C Álvarez-Esteban, E Del Barrio, JA Cuesta-Albertos, and C Matrán. "A fixed-point approach to barycenters in Wasserstein space". In: *Journal of Mathematical Analysis and Applications* 441.2 (2016), pp. 744–762.

[AM23] Elham Afzali and Saman Muthukumarana. "Gradient-Free Kernel Conditional Stein Discrepancy goodness of fit testing". In: *Machine Learning with Applications* 12 (2023), p. 100463. ISSN: 2666-8270. DOI: https://doi.org/10.1016/j.mlwa.2023.100463. URL: https://www.sciencedirect.com/science/article/pii/S2666827023000166.

[And+16] Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando De Freitas. "Learning to learn by gradient descent by gradient descent". In: *Advances in neural information processing systems*. 2016, pp. 3981–3989.

[AP07] Yves F Atchadé and François Perron. "On the geometric ergodicity of Metropolis-Hastings algorithms". In: *Statistics* 41.1 (2007), pp. 77–84.

[Aro50] Nachman Aronszajn. "Theory of reproducing kernels". In: *Transactions of the American mathematical society* 68.3 (1950), pp. 337–404.

[ASM21] David Alvarez-Melis, Yair Schiff, and Youssef Mroueh. "Optimizing functionals on the space of probabilities with input convex neural networks". In: *arXiv preprint arXiv:2106.00774* (2021).

[AV22] Michael S Albergo and Eric Vanden-Eijnden. "Building normalizing flows with stochastic interpolants". In: *arXiv preprint arXiv:2209.15571* (2022).

[AXK17] Brandon Amos, Lei Xu, and J Zico Kolter. "Input convex neural networks". In: *International Conference on Machine Learning*. PMLR. 2017, pp. 146–155.

[Bal+22] Krishna Balasubramanian, Sinho Chewi, Murat A Erdogdu, Adil Salim, and Shunshi Zhang. "Towards a theory of non-log-concave sampling: first-order stationarity guarantees for Langevin Monte Carlo". In: *Conference on Learning Theory*. PMLR. 2022, pp. 2896–2923.

[Bar+19] Alessandro Barp, Francois-Xavier Briol, Andrew Duncan, Mark Girolami, and Lester Mackey. "Minimum stein discrepancy estimators". In: *Advances in Neural Information Processing Systems* 32 (2019).

[Bar+22] Alessandro Barp, Carl-Johann Simon-Gabriel, Mark Girolami, and Lester Mackey. "Targeted separation and convergence with kernel discrepancies". In: *NeurIPS 2022 Workshop on Score-Based Methods*. 2022.

[BCW10] Martin Burger, José A Carrillo, and Marie-Therese Wolfram. "A mixed finite element method for nonlinear diffusion equations". In: *Kinetic & Related Models* 3.1 (2010), p. 59.

[BEB07] R Brits, Andries Petrus Engelbrecht, and Frans van den Bergh. "Locating multiple optima using particle swarm optimization". In: *Applied Mathematics and Computation* 189.2 (2007), pp. 1859–1883.

[Ben+15] Jean-David Benamou, Guillaume Carlier, Marco Cuturi, Luca Nenna, and Gabriel Peyré. "Iterative Bregman projections for regularized transportation problems". In: *SIAM Journal on Scientific Computing* 37.2 (2015), A1111–A1138.

[Ben+16] Jean-David Benamou, Guillaume Carlier, Quentin Mérigot, and Edouard Oudet. "Discretization of functionals involving the Monge–Ampère operator". In: *Numerische mathematik* 134.3 (2016), pp. 611–636.

[BG13] Simon Byrne and Mark Girolami. "Geodesic Monte Carlo on embedded manifolds". In: *Scandinavian Journal of Statistics* 40.4 (2013), pp. 825–845.

[BHH21] Pierre Blanchard, Desmond J Higham, and Nicholas J Higham. "Accurately computing the log-sum-exp and softmax functions". In: *IMA Journal of Numerical Analysis* 41.4 (2021), pp. 2311–2330.

[BHS19] Sergiy V Borodachov, Douglas P Hardin, and Edward B Saff. *Discrete energy on rectifiable sets*. Springer, 2019.

[Bil+21] Marin Biloš, Johanna Sommer, Syama Sundar Rangapuram, Tim Januschowski, and Stephan Günnemann. "Neural Flows: Efficient Alternative to Neural ODEs". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 21325–21337.

[Bil13] Patrick Billingsley. *Convergence of probability measures*. John Wiley & Sons, 2013.

[BKM17] David M Blei, Alp Kucukelbir, and Jon D McAuliffe. "Variational inference: A review for statisticians". In: *Journal of the American statistical Association* 112.518 (2017), pp. 859–877.

[Blo+22] Mathieu Blondel, Quentin Berthet, Marco Cuturi, Roy Frostig, Stephan Hoyer, Felipe Llinares-López, Fabian Pedregosa, and Jean-Philippe Vert. "Efficient and modular implicit differentiation". In: *Advances in neural information processing systems* 35 (2022), pp. 5230–5242.

[BMZ02] Samuel Burer, Renato DC Monteiro, and Yin Zhang. "Rank-two relaxation heuristics for max-cut and other binary quadratic programs". In: *SIAM Journal on Optimization* 12.2 (2002), pp. 503–521.

[Bon+11] Nicolas Bonneel, Michiel Van De Panne, Sylvain Paris, and Wolfgang Heidrich. "Displacement interpolation using Lagrangian mass transport". In: *Proceedings of the 2011 SIGGRAPH Asia conference*. 2011, pp. 1–12.

[Bra+18] James Bradbury et al. *JAX: composable transformations of Python+NumPy programs*. Version 0.3.13. 2018. URL: http://github.com/google/jax.

[Bra05] Richard C Bradley. "Basic properties of strong mixing conditions. A survey and some open questions". In: (2005).

[Bro+11] Steve Brooks, Andrew Gelman, Galin Jones, and Xiao-Li Meng. *Handbook of markov chain monte carlo*. CRC press, 2011.

[BSD24] Clément Bénard, Brian Staber, and Sébastien Da Veiga. "Kernel Stein Discrepancy thinning: a theoretical perspective of pathologies and a practical fix with regularization". In: *Advances in Neural Information Processing Systems* 36 (2024).

[BSU12] Marcus Brubaker, Mathieu Salzmann, and Raquel Urtasun. "A family of MCMC methods on implicitly defined manifolds". In: *Artificial intelligence and statistics*. PMLR. 2012, pp. 161–172.

[BT09] Amir Beck and Marc Teboulle. "A fast iterative shrinkage-thresholding algorithm for linear inverse problems". In: *SIAM journal on imaging sciences* 2.1 (2009), pp. 183–202.

[Bun+22] Charlotte Bunne, Laetitia Papaxanthos, Andreas Krause, and Marco Cuturi. "Proximal Optimal Transport Modeling of Population Dynamics". In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2022, pp. 6511–6528.

[BV23] Nicholas M Boffi and Eric Vanden-Eijnden. "Probability flow solution of the Fokker–Planck equation". In: *Machine Learning: Science and Technology* 4.3 (July 2023), p. 035012. DOI: 10.1088/2632-2153/ace2aa. URL: https://dx.doi.org/10.1088/2632-2153/ace2aa.

[Cao+19] Yue Cao, Tianlong Chen, Zhangyang Wang, and Yang Shen. "Learning to optimize in swarms". In: *Advances in neural information processing systems* 32 (2019).

[Cao+24] Hanqun Cao, Cheng Tan, Zhangyang Gao, Yilun Xu, Guangyong Chen, Pheng-Ann Heng, and Stan Z Li. "A survey on generative diffusion models". In: *IEEE Transactions on Knowledge and Data Engineering* (2024).

[Car+17] Bob Carpenter, Andrew Gelman, Matthew D Hoffman, Daniel Lee, Ben Goodrich, Michael Betancourt, Marcus Brubaker, Jiqiang Guo, Peter Li, and Allen Riddell. "Stan: A probabilistic programming language". In: *Journal of Statistical Software* 76.1 (2017).

[Car+20] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. "End-to-end object detection with transformers". In: *European Conference on Computer Vision*. Springer. 2020, pp. 213–229.

[CB19] Trevor Campbell and Tamara Broderick. "Automated scalable Bayesian inference via Hilbert coresets". In: *The Journal of Machine Learning Research* 20.1 (2019), pp. 551–588.

[CC96] Mary Kathryn Cowles and Bradley P Carlin. "Markov chain Monte Carlo convergence diagnostics: a comparative review". In: *Journal of the American Statistical Association* 91.434 (1996), pp. 883–904.

[CCH14] José Antonio Carrillo, Young-Pil Choi, and Maxime Hauray. "The derivation of swarming models: mean-field limit and Wasserstein distances". In: *Collective dynamics from bacteria to crowds*. Springer, 2014, pp. 1–46.

[CCH15] José A Carrillo, Alina Chertock, and Yanghong Huang. "A finite-volume method for nonlinear nonlocal equations with a gradient flow structure". In: *Communications in Computational Physics* 17.1 (2015), pp. 233–258.

[CCS18] Sebastian Claici, Edward Chien, and Justin Solomon. "Stochastic Wasserstein barycenters". In: *arXiv:1802.05757* (2018).

[CD14] Marco Cuturi and Arnaud Doucet. "Fast computation of Wasserstein barycenters". In: *Journal of Machine Learning Research* (2014).

[CD21] Nicolas Chopin and Gabriel Ducrocq. "Fast compression of MCMC output". In: *Entropy* 23.8 (2021), p. 1017.

[CDT06] Claudio Carmeli, Ernesto De Vito, and Alessandro Toigo. "Vector valued reproducing kernel Hilbert spaces of integrable functions and Mercer theorem". In: *Analysis and Applications* 4.04 (2006), pp. 377–408.

[CFT14] Nicolas Courty, Rémi Flamary, and Devis Tuia. "Domain adaptation with regularized optimal transport". In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer. 2014, pp. 274–289.

[CG14] Laming Chen and Yuantao Gu. "The convergence guarantees of a non-convex approach for sparse recovery". In: *IEEE Transactions on Signal Processing* 62.15 (2014), pp. 3754–3767.

[Cha15] Tapas Kumar Chandra. "De La Vallée Poussin's theorem, uniform integrability, tightness and moments". In: *Statistics & Probability Letters* 107 (2015), pp. 136–141.

[Che+17] Yutian Chen, Matthew W Hoffman, Sergio Gómez Colmenarejo, Misha Denil, Timothy P Lillicrap, Matt Botvinick, and Nando Freitas. "Learning to learn without gradient descent by gradient descent". In: *International Conference on Machine Learning*. PMLR. 2017, pp. 748–756.

[Che+18] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. "Neural ordinary differential equations". In: *Advances in neural information processing systems* 31 (2018).

[Che+20] Sinho Chewi, Thibaut Le Gouic, Chen Lu, Tyler Maunu, and Philippe Rigollet. "SVGD as a kernelized Wasserstein gradient flow of the chi-squared divergence". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 2098–2109.

[Che+22a] Tianlong Chen, Xiaohan Chen, Wuyang Chen, Howard Heaton, Jialin Liu, Zhangyang Wang, and Wotao Yin. "Learning to optimize: A primer and a benchmark". In: *Journal of Machine Learning Research* 23.189 (2022), pp. 1–59.

[Che+22b] Yifan Chen, Ethan N Epperly, Joel A Tropp, and Robert J Webber. "Randomly pivoted Cholesky: Practical approximation of a kernel matrix with few entry evaluations". In: *arXiv preprint arXiv:2207.06503* (2022).

[Che+23] Shoufa Chen, Peize Sun, Yibing Song, and Ping Luo. "Diffusiondet: Diffusion model for object detection". In: (2023), pp. 19830–19843.

[Che17] Yen-Chi Chen. "A tutorial on kernel density estimation and recent advances". In: *Biostatistics & Epidemiology* 1.1 (2017), pp. 161–187.

[Chi22] Lenaic Chizat. "Sparse optimization on measures with over-parameterized gradient descent". In: *Mathematical Programming* 194.1 (2022), pp. 487–532.

[CL99] Dan Crisan and Terry Lyons. "A particle approximation of the solution of the Kushner–Stratonovitch equation". In: *Probability Theory and Related Fields* 115.4 (1999), pp. 549–578.

[Cla+19] Christian Clason, Dirk A Lorenz, Hinrich Mahler, and Benedikt Wirth. "Entropic regularization of continuous optimal transport problems". In: *arXiv:1906.01333* (2019).

[CP16] Marco Cuturi and Gabriel Peyré. "A smoothed dual approach for variational Wasserstein problems". In: *SIAM Journal on Imaging Sciences* 9.1 (2016), pp. 320–343.

[Cra+23] Katy Craig, Karthik Elamvazhuthi, Matt Haberland, and Olga Turanova. "A blob method for inhomogeneous diffusion with applications to multi-agent control and sampling". In: *Mathematics of Computation* 92.344 (2023), pp. 2575–2654.

[CS08] Rick Chartrand and Valentina Staneva. "Restricted isometry properties and nonconvex compressive sensing". In: *Inverse Problems* 24.3 (2008), p. 035020.

[CSG16] Kacper Chwialkowski, Heiko Strathmann, and Arthur Gretton. "A Kernel Test of Goodness of Fit". In: *Proceedings of The 33rd International Conference on Machine Learning*. Ed. by Maria Florina Balcan and Kilian Q. Weinberger. Vol. 48. Proceedings of Machine Learning Research. New York, New York, USA: PMLR, June 2016, pp. 2606–2615. URL: https://proceedings.mlr.press/v48/chwialkowski16.html.

[CSX13] Wenfei Cao, Jian Sun, and Zongben Xu. "Fast image deconvolution using closed-form thresholding formulas of lq (q= 12, 23) regularization". In: *Journal of visual communication and image representation* 24.1 (2013), pp. 31–41.

[CTZ20] Zhiqin Chen, Andrea Tagliasacchi, and Hao Zhang. "Bsp-net: Generating compact meshes via binary space partitioning". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 45–54.

[Cut+22] Marco Cuturi, Laetitia Meng-Papaxanthos, Yingtao Tian, Charlotte Bunne, Geoff Davis, and Olivier Teboul. "Optimal transport tools (OTT): A jax toolbox for all things Wasserstein". In: *arXiv preprint arXiv:2201.12324* (2022).

[Cut13] Marco Cuturi. "Sinkhorn distances: Lightspeed computation of optimal transport". In: *Advances in Neural Information Processing Systems*. 2013, pp. 2292–2300.

[Dax+14] Achiya Dax et al. "Low-rank positive approximants of symmetric matrices". In: *Advances in Linear Algebra & Matrix Theory* 4.03 (2014), p. 172.

[DC05] Randal Douc and Olivier Cappé. "Comparison of resampling schemes for particle filtering". In: *ISPA 2005. Proceedings of the 4th International Symposium on Image and Signal Processing and Analysis, 2005.* IEEE. 2005, pp. 64–69.

[Den+20] Boyang Deng, Kyle Genova, Soroosh Yazdani, Sofien Bouaziz, Geoffrey Hinton, and Andrea Tagliasacchi. "Cvxnet: Learnable convex decomposition". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 31–44.

[DG19] Damek Davis and Benjamin Grimmer. "Proximally guided stochastic subgradient method for nonsmooth, nonconvex problems". In: *SIAM Journal on Optimization* 29.3 (2019), pp. 1908–1930.

[DM00] Angel-Victor DeMiguel and Walter Murray. "An analysis of collaborative optimization methods". In: *8th symposium on multidisciplinary analysis and optimization*. 2000, p. 4720.

[DM17] Alain Durmus and Eric Moulines. "Nonasymptotic convergence analysis for the unadjusted Langevin algorithm". In: (2017).

[DM21] Raaz Dwivedi and Lester Mackey. "Kernel thinning". In: *arXiv preprint arXiv:2105.05842* (2021).

[DM22a] Alain Durmus and Éric Moulines. "On the geometric convergence for MALA under verifiable conditions". In: *arXiv preprint arXiv:2201.01951* (2022).

[DM22b] Raaz Dwivedi and Lester Mackey. "Generalized Kernel Thinning". In: *International Conference on Learning Representations*. 2022.

[DMM19] Alain Durmus, Szymon Majewski, and Błażej Miasojedow. "Analysis of Langevin Monte Carlo via convex optimization". In: *The Journal of Machine Learning Research* 20.1 (2019), pp. 2666–2711.

[DMS20] Alain Durmus, Éric Moulines, and Eero Saksman. "Irreducibility and geometric ergodicity of Hamiltonian Monte Carlo". In: *The Annals of Statistics* 48.6 (2020), pp. 3545–3564.

[Dou+18] Randal Douc, Eric Moulines, Pierre Priouret, and Philippe Soulier. *Markov chains*. Vol. 1. Springer, 2018.

[DSB16] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. "Density estimation using real nvp". In: *arXiv preprint arXiv:1605.08803* (2016).

[Dvu+18] Pavel Dvurechenskii, Darina Dvinskikh, Alexander Gasnikov, Cesar Uribe, and Angelia Nedich. "Decentralize and randomize: Faster algorithm for Wasserstein barycenters". In: *Advances in Neural Information Processing Systems*. 2018, pp. 10760–10770.

[EM12] Tarek A El Moselhy and Youssef M Marzouk. "Bayesian inference with optimal maps". In: *Journal of Computational Physics* 231.23 (2012), pp. 7815–7850.

[EM24] Ethan Epperly and Elvira Moreno. "Kernel quadrature with randomly pivoted cholesky". In: *Advances in Neural Information Processing Systems* 36 (2024).

[EMS18] Murat A Erdogdu, Lester Mackey, and Ohad Shamir. "Global non-convex optimization with discretized diffusions". In: *Advances in Neural Information Processing Systems* 31 (2018).

[ET99] Ivar Ekeland and Roger Temam. *Convex Analysis and Variational Problems*. Vol. 28. SIAM, 1999.

[EUD18] Stefan Elfwing, Eiji Uchibe, and Kenji Doya. "Sigmoid-weighted linear units for neural network function approximation in reinforcement learning". In: *Neural Networks* 107 (2018), pp. 3–11.

[Fan+23] Zhenghan Fang, Kuo-Wei Lai, Peter van Zijl, Xu Li, and Jeremias Sulam. "DeepSTI: towards tensor reconstruction using fewer orientations in susceptibility tensor imaging". In: *Medical image analysis* 87 (2023), p. 102829.

[Fey+19] Jean Feydy, Thibault Séjourné, François-Xavier Vialard, Shun-ichi Amari, Alain Trouvé, and Gabriel Peyré. "Interpolating between optimal transport and mmd using sinkhorn divergences". In: *The 22nd International Conference on Artificial Intelligence and Statistics*. PMLR. 2019, pp. 2681–2690.

[FG15] Nicolas Fournier and Arnaud Guillin. "On the rate of convergence in Wasserstein distance of the empirical measure". In: *Probability Theory and Related Fields* 162.3-4 (2015), pp. 707–738.

[Fri44] Kurt Otto Friedrichs. "The identity of weak and strong extensions of differential operators". In: *Transactions of the American Mathematical Society* 55.1 (1944), pp. 132–151.

[FTC21] Jiaojiao Fan, Amirhossein Taghvaei, and Yongxin Chen. "Variational Wasserstein gradient flow". In: *arXiv preprint arXiv:2112.02424* (2021).

[Gao+20] Lin Gao, Ling-Xiao Zhang, Hsien-Yu Meng, Yi-Hui Ren, Yu-Kun Lai, and Leif Kobbelt. "PRS-Net: Planar reflective symmetry detection net for 3D models". In: *IEEE Transactions on Visualization and Computer Graphics* 27.6 (2020), pp. 3007–3018.

[Gen+16] Aude Genevay, Marco Cuturi, Gabriel Peyré, and Francis Bach. "Stochastic optimization for large-scale optimal transport". In: *Advances in Neural Information Processing Systems*. 2016, pp. 3440–3448.

[Gho+21] Benyamin Ghojogh, Ali Ghodsi, Fakhri Karray, and Mark Crowley. "KKT conditions, first-order and second-order optimization, and distributed optimization: tutorial and survey". In: *arXiv preprint arXiv:2110.01858* (2021).

[GJZ17] Rong Ge, Chi Jin, and Yi Zheng. "No spurious local minima in nonconvex low rank problems: A unified geometric analysis". In: *International Conference on Machine Learning*. PMLR. 2017, pp. 1233–1242.

[GL10] Karol Gregor and Yann LeCun. "Learning fast approximations of sparse coding". In: *Proceedings of the 27th International Conference on Machine Learning*. 2010, pp. 399–406.

[GL21] Chengyue Gong and Xingchao Liu. "Bi-objective Trade-off with Dynamic Barrier Gradient Descent". In: *NeurIPS 2021* (2021).

[GLR23] M. A. Gallegos-Herrada, D. Ledvinka, and J. S. Rosenthal. *Equivalences of Geometric Ergodicity of Markov Chains*. 2023. arXiv: 2203.04395 [math.PR].

[GM17] Jackson Gorham and Lester Mackey. "Measuring sample quality with kernels". In: *International Conference on Machine Learning*. PMLR. 2017, pp. 1292–1301.

[Gor+19] Jackson Gorham, Andrew B Duncan, Sebastian J Vollmer, and Lester Mackey. "Measuring sample quality with diffusions". In: *The Annals of Applied Probability* 29.5 (2019), pp. 2884–2928.

[GOW21] Davis Gilton, Gregory Ongie, and Rebecca Willett. "Deep equilibrium architectures for inverse problems in imaging". In: *IEEE Transactions on Computational Imaging* 7 (2021), pp. 1123–1133.

[GPC18] Aude Genevay, Gabriel Peyré, and Marco Cuturi. "Learning generative models with sinkhorn divergences". In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2018, pp. 1608–1617.

[Gre+12] Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. "A kernel two-sample test". In: *The Journal of Machine Learning Research* 13.1 (2012), pp. 723–773.

[GSK10] Robert Gramacy, Richard Samworth, and Ruth King. "Importance tempering". In: *Statistics and Computing* 20 (2010), pp. 1–7.

[GW95] Michel X Goemans and David P Williamson. "Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming". In: *Journal of the ACM (JACM)* 42.6 (1995), pp. 1115–1145.

[HD05] Aapo Hyvärinen and Peter Dayan. "Estimation of non-normalized statistical models by score matching." In: *Journal of Machine Learning Research* 6.4 (2005).

[He+16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[Hee+20] Jonathan Heek, Anselm Levskaya, Avital Oliver, Marvin Ritter, Bertrand Rondepierre, Andreas Steiner, and Marc van Zee. *Flax: A neural network library and ecosystem for JAX*. Version 0.6.3. 2020. URL: http://github.com/google/flax.

[HG+14] Matthew D Hoffman, Andrew Gelman, et al. "The No-U-Turn sampler: adaptively setting path lengths in Hamiltonian Monte Carlo." In: *J. Mach. Learn. Res.* 15.1 (2014), pp. 1593–1623.

[HKZ22] Cole Hawkins, Alec Koppel, and Zheng Zhang. "Online, informative mcmc thinning with kernelized stein discrepancy". In: *arXiv preprint arXiv:2201.07130* (2022).

[HLO20] Tim Hoheisel, Maxime Laborde, and Adam Oberman. "A regularization interpretation of the proximal point method for weakly convex functions". In: *Journal of Dynamics & Games* 7.1 (2020), p. 79.

[HOL22] Satoshi Hayakawa, Harald Oberhauser, and Terry Lyons. "Positively weighted kernel quadrature via subsampling". In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 6886–6900.

[HOL23] Satoshi Hayakawa, Harald Oberhauser, and Terry Lyons. "Sampling-based Nyström approximation and kernel quadrature". In: *Proceedings of the 40th International Conference on Machine Learning*. ICML'23. Honolulu, Hawaii, USA: JMLR.org, 2023.

[Hör15] Lars Hörmander. *The analysis of linear partial differential operators I: Distribution theory and Fourier analysis*. Springer, 2015.

[HSR20] Liam Hodgkinson, Robert Salomone, and Fred Roosta. "The reproducing Stein kernel approach for post-hoc corrected sampling". In: *arXiv preprint arXiv:2001.09266* (2020).

[JKO98] Richard Jordan, David Kinderlehrer, and Felix Otto. "The variational formulation of the Fokker–Planck equation". In: *SIAM journal on mathematical analysis* 29.1 (1998), pp. 1–17.

[Joh09] Alicia A Johnson. *Geometric ergodicity of Gibbs samplers*. university of minnesota, 2009.

[Jos+19] V Roshan Joseph, Dianpeng Wang, Li Gu, Shiji Lyu, and Rui Tuo. "Deterministic sampling of expensive posteriors using minimum energy designs". In: *Technometrics* (2019).

[Kar+21] George Em Karniadakis, Ioannis G Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang. "Physics-informed machine learning". In: *Nature Reviews Physics* 3.6 (2021), pp. 422–440.

[Kar+22] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. "Elucidating the design space of diffusion-based generative models". In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 26565–26577.

[KB14] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[KH19] Kenji Kawaguchi and Jiaoyang Huang. "Gradient descent finds global minima for generalizable deep neural networks of practical sizes". In: *2019 57th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE. 2019, pp. 92–99.

[Kid21] Patrick Kidger. "On Neural Differential Equations". PhD thesis. University of Oxford, 2021.

[Kim+20] Sangpil Kim, Hyung-gun Chi, Xiao Hu, Qixing Huang, and Karthik Ramani. "A Large-scale Annotated Mechanical Components Benchmark for Classification and Retrieval Tasks with Deep Neural Networks". In: *Proceedings of 16th European Conference on Computer Vision (ECCV)*. 2020.

[Kle13] Achim Klenke. *Probability theory: a comprehensive course*. Springer Science & Business Media, 2013.

[Koh+96] Ron Kohavi et al. "Scaling up the accuracy of naive-bayes classifiers: A decision-tree hybrid." In: *Kdd*. Vol. 96. 1996, pp. 202–207.

[Kor+20] Anna Korba, Adil Salim, Michael Arbel, Giulia Luise, and Arthur Gretton. "A non-asymptotic analysis for Stein variational gradient descent". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 4672–4682.

[Kor+21a] Anna Korba, Pierre-Cyril Aubin-Frankowski, Szymon Majewski, and Pierre Ablin. "Kernel stein discrepancy descent". In: *International Conference on Machine Learning*. PMLR. 2021, pp. 5719–5730.

[Kor+21b] Alexander Korotin, Lingxiao Li, Justin Solomon, and Evgeny Burnaev. "Continuous Wasserstein-2 Barycenter Estimation without Minimax Optimization". In: *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL: https://openreview.net/forum?id=3tFAs5E-Pe.

[Kor+22] Alexander Korotin, Vage Egiazarian, Lingxiao Li, and Evgeny Burnaev. "Wasserstein iterative networks for barycenter estimation". In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 15672–15686.

[KSS21] Alexey Kroshnin, Vladimir Spokoiny, and Alexandra Suvorikova. "Statistical inference for Bures–Wasserstein barycenters". In: *The Annals of Applied Probability* 31.3 (2021), pp. 1264–1298.

[Küh16] Franziska Kühn. "Probability and heat kernel estimates for Lévy (-type) Processes". In: (2016).

[LDG23] Xing Liu, Andrew B. Duncan, and Axel Gandy. "Using Perturbation to Improve Goodness-of-Fit Tests based on Kernelized Stein Discrepancy". In: *Proceedings of the 40th International Conference on Machine Learning*. Ed. by Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett. Vol. 202. Proceedings of Machine Learning Research. PMLR, July 2023, pp. 21527–21547.

[LDM24] Lingxiao Li, Raaz Dwivedi, and Lester Mackey. "Debiased distribution compression". In: 0 (2024).

[LFS21] Lauro Langosco di Langosco, Vincent Fortuin, and Heiko Strathmann. "Neural variational gradient descent". In: *arXiv preprint arXiv:2107.10731* (2021).

[LGL22] Xingchao Liu, Chengyue Gong, and Qiang Liu. "Flow straight and fast: Learning to generate and transfer data with rectified flow". In: *arXiv preprint arXiv:2209.03003* (2022).

[LHS20] Phuc H Le-Khac, Graham Healy, and Alan F Smeaton. "Contrastive representation learning: A framework and review". In: *Ieee Access* 8 (2020), pp. 193907–193934.

[LHS23] Lingxiao Li, Samuel Hurault, and Justin M Solomon. "Self-Consistent Velocity Matching of Probability Flows". In: *Advances in Neural Information Processing Systems*. Ed. by A. Oh, T. Neumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine. Vol. 36. Curran Associates, Inc., 2023, pp. 57038–57057. URL: https://proceedings.neurips.cc/paper_files/paper/2023/file/b2b781badeeb49896c4b324c466ec442-Paper-Conference.pdf.

[Li+16] Xiaodong Li, Michael G Epitropakis, Kalyanmoy Deb, and Andries Engelbrecht. "Seeking multiple solutions: An updated survey on niching methods and their applications". In: *IEEE Transactions on Evolutionary Computation* 21.4 (2016), pp. 518–538.

[Li+19] Lingxiao Li, Minhyuk Sung, Anastasia Dubrovina, Li Yi, and Leonidas J Guibas. "Supervised fitting of geometric primitives to 3d point clouds". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 2652–2660.

[Li+20] Lingxiao Li, Aude Genevay, Mikhail Yurochkin, and Justin Solomon. "Continuous regularized Wasserstein barycenters". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 17755–17765.

[Li+23a] Lingxiao Li, Noam Aigerman, Vladimir G. Kim, Jiajin Li, Kristjan H. Greenewald, Mikhail Yurochkin, and Justin Solomon. "Learning Proximal Operators to Discover Multiple Optima". In: *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL: https://openreview.net/pdf?id=PzBGIullo7.

[Li+23b] Lingxiao Li, Qiang Liu, Anna Korba, Mikhail Yurochkin, and Justin Solomon. "Sampling with Mollified Interaction Energy Descent". In: *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL: https://openreview.net/pdf?id=zWy7dqOcel.

[Li09] Xiaodong Li. "Niching without niching parameters: particle swarm optimization using a ring topology". In: *IEEE Transactions on Evolutionary Computation* 14.1 (2009), pp. 150–169.

[Lin+14] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. "Microsoft coco: Common objects in context". In: *European conference on computer vision*. Springer. 2014, pp. 740–755.

[Lip+22] Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. "Flow matching for generative modeling". In: *arXiv preprint arXiv:2210.02747* (2022).

[Liu+20] Li Liu, Wanli Ouyang, Xiaogang Wang, Paul Fieguth, Jie Chen, Xinwang Liu, and Matti Pietikäinen. "Deep learning for generic object detection: A survey". In: *International journal of computer vision* 128.2 (2020), pp. 261–318.

[Liu+21] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. "Swin transformer: Hierarchical vision transformer using shifted windows". In: (2021), pp. 10012–10022.

[Liu17] Qiang Liu. "Stein variational gradient descent as gradient flow". In: *Advances in neural information processing systems* 30 (2017).

[LL17] Qiang Liu and Jason Lee. "Black-box importance sampling". In: *Artificial Intelligence and Statistics*. PMLR. 2017, pp. 952–961.

[LLJ16] Qiang Liu, Jason Lee, and Michael Jordan. "A kernelized Stein discrepancy for goodness-of-fit tests". In: *International conference on machine learning*. PMLR. 2016, pp. 276–284.

[LLN19] Yulong Lu, Jianfeng Lu, and James Nolen. "Accelerating langevin sampling with birth-death". In: *arXiv preprint arXiv:1905.09863* (2019).

[LLW23] Lei Li, Jian-Guo Liu, and Yuliang Wang. "Geometric ergodicity of SGLD via reflection coupling". In: *arXiv preprint arXiv:2301.06769* (2023).

[LM16] Ke Li and Jitendra Malik. "Learning to optimize". In: *arXiv preprint arXiv:1606.01885* (2016).

[LMM19] Dirk A Lorenz, Paul Manns, and Christian Meyer. "Quadratically regularized optimal transport". In: *Applied Mathematics & Optimization* (2019), pp. 1–31.

[LTL21] Xingchao Liu, Xin Tong, and Qiang Liu. "Sampling with Trusthworthy Constraints: A Variational Gradient Framework". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 23557–23568.

[Lui+19] Giulia Luise, Saverio Salzo, Massimiliano Pontil, and Carlo Ciliberto. "Sinkhorn Barycenters with Free Support via Frank-Wolfe Algorithm". In: *Advances in Neural Information Processing Systems*. 2019, pp. 9318–9329.

[LW16] Qiang Liu and Dilin Wang. "Stein variational gradient descent: A general purpose bayesian inference algorithm". In: *Advances in neural information processing systems* 29 (2016).

[LW18] Jeffrey Larson and Stefan M Wild. "Asynchronously parallel optimization solver for finding multiple minima". In: *Mathematical Programming Computation* 10.3 (2018), pp. 303–332.

[LXY13] Ming-Jun Lai, Yangyang Xu, and Wotao Yin. "Improved iteratively reweighted least squares for unconstrained smoothed \ell_q minimization". In: *SIAM Journal on Numerical Analysis* 51.2 (2013), pp. 927–957.

[LZ18] Chang Liu and Jun Zhu. "Riemannian Stein variational gradient descent for Bayesian inference". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 32. 1. 2018.

[Mar+18] Elaine Crespo Marques, Nilson Maciel, Lirida Naviner, Hao Cai, and Jun Yang. "A review of sparse recovery algorithms". In: *IEEE access* 7 (2018), pp. 1300–1322.

[Mei+17] Tim Meinhardt, Michael Moller, Caner Hazirbas, and Daniel Cremers. "Learning proximal operators: Using denoising networks for regularizing inverse imaging problems". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 1781–1790.

[Min+14] Stanislav Minsker, Sanvesh Srivastava, Lizhen Lin, and David Dunson. "Scalable and robust Bayesian inference via the median posterior". In: *International Conference on Machine Learning*. 2014, pp. 1656–1664.

[Min10] Ha Quang Minh. "Some properties of Gaussian reproducing kernel Hilbert spaces and their implications for function approximation and learning theory". In: *Constructive Approximation* 32 (2010), pp. 307–338.

[Mit+13] Niloy J Mitra, Mark Pauly, Michael Wand, and Duygu Ceylan. "Symmetry in 3d geometry: Extraction and applications". In: *Computer Graphics Forum*. Vol. 32. 6. Wiley Online Library. 2013, pp. 1–23.

[Mok+21] Petr Mokrov, Alexander Korotin, Lingxiao Li, Aude Genevay, Justin Solomon, and Evgeny Burnaev. "Large-scale Wasserstein gradient flows". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 15243–15256.

[Mor62] Jean Jacques Moreau. "Fonctions convexes duales et points proximaux dans un espace hilbertien". In: *Comptes rendus hebdomadaires des séances de l'Académie des sciences* 255 (1962), pp. 2897–2899.

[MPU97] Florence Merlevède, Magda Peligrad, and Sergey Utev. "Sharp conditions for the CLT of linear processes in a Hilbert space". In: *Journal of Theoretical Probability* 10.3 (1997), pp. 681–693.

[MT12] Sean P Meyn and Richard L Tweedie. *Markov chains and stochastic stability*. Springer Science & Business Media, 2012.

[Mus+02] Ken Museth, David E Breen, Ross T Whitaker, and Alan H Barr. "Level set surface editing operators". In: *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*. 2002, pp. 330–338.

[Nie+11] Steven Niederer, Lawrence Mitchell, Nicolas Smith, and Gernot Plank. "Simulating human cardiac electrophysiology on clinical time-scales". In: *Frontiers in physiology* 2 (2011), p. 14.

[NV06] Alexander Neubeck and Luc Van Gool. "Efficient non-maximum suppression". In: *18th International Conference on Pattern Recognition (ICPR'06)*. Vol. 3. IEEE. 2006, pp. 850–855.

[NZE05] Patrick Ngatchou, Anahita Zarei, and A El-Sharkawi. "Pareto multi objective optimization". In: *Proceedings of the 13th international conference on, intelligent systems application to power systems*. IEEE. 2005, pp. 84–91.

[Pan+19] Zhaoqing Pan, Weijie Yu, Xiaokai Yi, Asifullah Khan, Feng Yuan, and Yuhui Zheng. "Recent progress on generative adversarial networks (GANs): A survey". In: *IEEE access* 7 (2019), pp. 36322–36333.

[Par+19] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. "Deepsdf: Learning continuous signed distance functions for shape representation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 165–174.

[Par+23] Min Sue Park, Cheolhyeong Kim, Hwijae Son, and Hyung Ju Hwang. "The deep minimizing movement scheme". In: *Journal of Computational Physics* 494 (2023), p. 112518.

[PB14] Neal Parikh and Stephen Boyd. "Proximal algorithms". In: *Foundations and Trends in optimization* 1.3 (2014), pp. 127–239.

[PC19] Gabriel Peyré and Marco Cuturi. "Computational optimal transport". In: *Foundations and Trends in Machine Learning* 11.5-6 (2019), pp. 355–607.

[Pey15] Gabriel Peyré. "Entropic approximation of Wasserstein gradient flows". In: *SIAM Journal on Imaging Sciences* 8.4 (2015), pp. 2323–2351.

[PFS21] Ioannis PA Papadopoulos, Patrick E Farrell, and Thomas M Surowiec. "Computing multiple solutions of topology optimization problems". In: *SIAM Journal on Scientific Computing* 43.3 (2021), A1555–A1582.

[Pin20] Iosif Pinelis. "Exact lower and upper bounds on the incomplete gamma function". In: *arXiv preprint arXiv:2005.06384* (2020).

[Pit17] Yannik Pitcan. "A note on concentration inequalities for U-statistics". In: *arXiv preprint arXiv:1712.06160* (2017).

[PKB07] Riccardo Poli, James Kennedy, and Tim Blackwell. "Particle swarm optimization: An overview". In: *Swarm intelligence* 1 (2007), pp. 33–57.

[Pod+06] Joshua Podolak, Philip Shilane, Aleksey Golovinskiy, Szymon Rusinkiewicz, and Thomas Funkhouser. "A Planar-Reflective Symmetry Transform for 3D Shapes". In: *ACM Trans. Graph.* 25.3 (July 2006), pp. 549–559. ISSN: 0730-0301. DOI: 10.1145/1141911.1141923. URL: https://doi.org/10.1145/1141911.1141923.

[PPJ19] Du Phan, Neeraj Pradhan, and Martin Jankowiak. "Composable Effects for Flexible and Accelerated Probabilistic Programming in NumPyro". In: *arXiv preprint arXiv:1912.11554* (2019).

[PR16] Vern I Paulsen and Mrinal Raghupathi. *An introduction to the theory of reproducing kernel Hilbert spaces*. Vol. 152. Cambridge university press, 2016.

[PZ21] Luc Pronzato and Anatoly Zhigljavsky. "Minimum-energy measures for singular kernels". In: *Journal of Computational and Applied Mathematics* 382 (2021), p. 113089.

[Qin23] Qian Qin. "Geometric ergodicity of trans-dimensional Markov chain Monte Carlo algorithms". In: *arXiv preprint arXiv:2308.00139* (2023).

[RDF78] Peter J Rossky, Jimmie D Doll, and Harold L Friedman. "Brownian dynamics as smart Monte Carlo simulation". In: *The Journal of Chemical Physics* 69.10 (1978), pp. 4628–4633.

[Ren+15] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. "Faster r-cnn: Towards real-time object detection with region proposal networks". In: *Advances in neural information processing systems* 28 (2015), pp. 91–99.

[Ria+20] Marina Riabiz, Wilson Ye Chen, Jon Cockayne, Pawel Swietach, Steven A. Niederer, Lester Mackey, and Chris J. Oates. *Replication Data for: Optimal Thinning of MCMC Output*. Version V2. Accessed on Mar 23, 2021. 2020. DOI: 10.7910/DVN/MDKNWM. URL: https://doi.org/10.7910/DVN/MDKNWM.

[Ria+22] Marina Riabiz, Wilson Ye Chen, Jon Cockayne, Pawel Swietach, Steven A Niederer, Lester Mackey, and Chris J Oates. "Optimal thinning of MCMC output". In: *Journal of the Royal Statistical Society Series B: Statistical Methodology* 84.4 (2022), pp. 1059–1081.

[Roc21] R Tyrrell Rockafellar. "Advances in convergence and scope of the proximal point algorithm". In: *J. Nonlinear and Convex Analysis* (2021).

[Roc76] R Tyrrell Rockafellar. "Monotone operators and the proximal point algorithm". In: *SIAM journal on control and optimization* 14.5 (1976), pp. 877–898.

[RPK19] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations". In: *Journal of Computational physics* 378 (2019), pp. 686–707.

[RR08] Ali Rahimi and Benjamin Recht. "Random features for large-scale kernel machines". In: *Advances in Neural Information Processing Systems*. 2008, pp. 1177–1184.

[RR96] Hannes Risken and Hannes Risken. *Fokker-planck equation*. Springer, 1996.

[RT96] Gareth O Roberts and Richard L Tweedie. "Geometric convergence and central limit theorems for multidimensional Hastings and Metropolis algorithms". In: *Biometrika* 83.1 (1996), pp. 95–110.

[San15] Filippo Santambrogio. *Optimal Transport for Applied Mathematicians*. Vol. 55. 58-63. Springer, 2015, p. 94.

[SBS23] Christopher Scarvelis, Haitz Sáez de Ocáriz Borde, and Justin Solomon. "Closed-form diffusion models". In: *arXiv preprint arXiv:2310.12395* (2023).

[SC08] Ingo Steinwart and Andreas Christmann. *Support vector machines*. Springer Science & Business Media, 2008.

[SDM22] Abhishek Shetty, Raaz Dwivedi, and Lester Mackey. "Distribution Compression in Near-linear Time". In: *International Conference on Learning Representations*. 2022.

[Seg+17] Vivien Seguy, Bharath Bhushan Damodaran, Rémi Flamary, Nicolas Courty, Antoine Rolet, and Mathieu Blondel. "Large-scale optimal transport and mapping estimation". In: *arXiv:1711.02283* (2017).

[Ser20] Sylvia Serfaty. "Mean field limit for Coulomb-type flows". In: *Duke Mathematical Journal* 169.15 (2020), pp. 2887–2935.

[She+22] Zebang Shen, Zhenfu Wang, Satyen Kale, Alejandro Ribeiro, Amin Karbasi, and Hamed Hassani. "Self-consistency of the fokker planck equation". In: *Conference on Learning Theory*. PMLR. 2022, pp. 817–841.

[Shi+20] Yifei Shi, Junwen Huang, Hongjia Zhang, Xin Xu, Szymon Rusinkiewicz, and Kai Xu. "SymmetryNet: learning to predict reflectional and rotational symmetries of 3D shapes from single-view RGB-D images". In: *ACM Transactions on Graphics (TOG)* 39.6 (2020), pp. 1–14.

[Sit+20] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. "Implicit neural representations with periodic activation functions". In: *Advances in neural information processing systems* 33 (2020), pp. 7462–7473.

[SLD18] Sanvesh Srivastava, Cheng Li, and David B Dunson. "Scalable Bayes via barycenter in Wasserstein space". In: *The Journal of Machine Learning Research* 19.1 (2018), pp. 312–346.

[SLM21] Jiaxin Shi, Chang Liu, and Lester Mackey. "Sampling with Mirrored Stein Operators". In: *arXiv preprint arXiv:2106.12506* (2021).

[Sol+14] Justin Solomon, Raif Rustamov, Leonidas Guibas, and Adrian Butscher. "Wasserstein propagation for semi-supervised learning". In: *International Conference on Machine Learning*. 2014, pp. 306–314.

[Sol+15] Justin Solomon, Fernando De Goes, Gabriel Peyré, Marco Cuturi, Adrian Butscher, Andy Nguyen, Tao Du, and Leonidas Guibas. "Convolutional Wasserstein distances: Efficient optimal transportation on geometric domains". In: *ACM Transactions on Graphics (TOG)* 34.4 (2015), pp. 1–11.

[Sor+04] Olga Sorkine, Daniel Cohen-Or, Yaron Lipman, Marc Alexa, Christian Rössl, and H-P Seidel. "Laplacian surface editing". In: *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*. 2004, pp. 175–184.

[SR13] Gábor J Székely and Maria L Rizzo. "Energy statistics: A class of statistics based on distances". In: *Journal of statistical planning and inference* 143.8 (2013), pp. 1249–1272.

[Sri+15] Sanvesh Srivastava, Volkan Cevher, Quoc Dinh, and David Dunson. "WASP: Scalable Bayes via barycenters of subset posteriors". In: *Artificial Intelligence and Statistics*. 2015, pp. 912–920.

[SS12] Ingo Steinwart and Clint Scovel. "Mercer's theorem on general domains: On the interaction between measures, kernels, and RKHSs". In: *Constructive Approximation* 35 (2012), pp. 363–417.

[SSR22] Adil Salim, Lukang Sun, and Peter Richtarik. "A Convergence Theory for SVGD in the Population Limit under Talagrand's Inequality T1". In: *International Conference on Machine Learning*. PMLR. 2022, pp. 19139–19152.

[Sta+17] Matthew Staib, Sebastian Claici, Justin M Solomon, and Stefanie Jegelka. "Parallel streaming Wasserstein barycenters". In: *Advances in Neural Information Processing Systems*. 2017, pp. 2647–2658.

[SW24] Zebang Shen and Zhenfu Wang. "Entropy-dissipation informed neural network for mckean-vlasov type pdes". In: *Advances in Neural Information Processing Systems* 36 (2024).

[SZ08] Hong-Wei Sun and Ding-Xuan Zhou. "Reproducing kernel Hilbert spaces associated with analytic translation-invariant Mercer kernels". In: *Journal of Fourier Analysis and Applications* 14.1 (2008), pp. 89–101.

[Tch16] Maria Tchernychova. "Caratheodory cubature measures". PhD thesis. University of Oxford, 2016.

[Tey+21] Onur Teymur, Jackson Gorham, Marina Riabiz, and Chris Oates. "Optimal quantisation of probability measures using maximum mean discrepancy". In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2021, pp. 1027–1035.

[Tib+10] Ryan Tibshirani et al. "Proximal gradient descent and acceleration". In: *Lecture Notes* (2010).

[Tib17] Ryan J Tibshirani. "Dykstra's algorithm, ADMM, and coordinate descent: Connections, insights, and extensions". In: *Advances in Neural Information Processing Systems* 30 (2017).

[TJ19] Amirhossein Taghvaei and Amin Jalali. "2-Wasserstein approximation via restricted convex potentials with application to improved training for GANs". In: *arXiv:1902.07197* (2019).

[TK07] Grigorios Tsoumakas and Ioannis Katakis. "Multi-label classification: An overview". In: *International Journal of Data Warehousing and Mining (IJDWM)* 3.3 (2007), pp. 1–13.

[TS20] Nicolas Garcia Trillos and Daniel Sanz-Alonso. "The Bayesian update: variational formulations and gradient flows". In: *Bayesian Analysis* 15.1 (2020), pp. 29–56.

[Var01] SR Srinivasa Varadhan. *Probability theory*. 7. American Mathematical Soc., 2001.

[Váz07] Juan Luis Vázquez. *The porous medium equation: mathematical theory*. Oxford University Press on Demand, 2007.

[Vil08] Cédric Villani. *Optimal Transport: Old and New*. Vol. 338. Springer Science & Business Media, 2008.

[VK21] Dootika Vats and Christina Knudson. "Revisiting the gelman–rubin diagnostic". In: *Statistical Science* 36.4 (2021), pp. 518–529.

[Wai19] Martin J Wainwright. *High-dimensional statistics: A non-asymptotic viewpoint*. Vol. 48. Cambridge university press, 2019.

[WAL23] Jun-Kun Wang, Jacob Abernethy, and Kfir Y Levy. "No-regret dynamics in the fenchel game: A unified framework for algorithmic convex optimization". In: *Mathematical Programming* (2023), pp. 1–66.

[Wan+19] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. "Dynamic graph cnn for learning on point clouds". In: *Acm Transactions On Graphics (tog)* 38.5 (2019), pp. 1–12.

[Wan+24] Congye Wang, Ye Chen, Heishiro Kanagawa, and Chris J Oates. "Stein Π-Importance Sampling". In: *Advances in Neural Information Processing Systems* 36 (2024).

[Wel+13] Jon Wellner et al. *Weak convergence and empirical processes: with applications to statistics*. Springer Science & Business Media, 2013.

[Wib18] Andre Wibisono. "Sampling as optimization in the space of measures: The Langevin dynamics as a composite optimization problem". In: *Conference on Learning Theory*. PMLR. 2018, pp. 2093–3027.

[WS19] Yue Wang and Justin M. Solomon. "PRNet: Self-Supervised Learning for Partial-to-Partial Registration". In: *33rd Conference on Neural Information Processing Systems*. 2019.

[WW10] Michael Westdickenberg and Jon Wilkening. "Variational particle schemes for the porous medium equation and for the system of isentropic Euler equations". In: *ESAIM: Mathematical Modelling and Numerical Analysis* 44.1 (2010), pp. 133–166.

[XKS22] Lantian Xu, Anna Korba, and Dejan Slepcev. "Accurate Quantization of Measures via Interacting Particle-based Optimization". In: *International Conference on Machine Learning*. PMLR. 2022, pp. 24576–24595.

[Xu21] Wenkai Xu. "Generalised Kernel Stein Discrepancy (GKSD): A Unifying Approach for Non-parametric Goodness-of-fit Testing". In: *arXiv preprint arXiv:2106.12105* (2021).

[Yan+18] Jiasen Yang, Qiang Liu, Vinayak Rao, and Jennifer Neville. "Goodness-of-fit testing for discrete distributions via Stein discrepancy". In: *International Conference on Machine Learning*. PMLR. 2018, pp. 5561–5570.

[Yan+20] Chengzhu Yang, Yuantao Gu, Badong Chen, Hongbing Ma, and Hing Cheung So. "Learning proximal operator methods for nonconvex sparse recovery with theoretical guarantee". In: *IEEE Transactions on Signal Processing* 68 (2020), pp. 5244–5259.

[Zha+18] Daokun Zhang, Jie Yin, Xingquan Zhu, and Chengqi Zhang. "Network representation learning: A survey". In: *IEEE transactions on Big Data* 6.1 (2018), pp. 3–28.

[Zha+20] Kelvin Shuangjian Zhang, Gabriel Peyré, Jalal Fadili, and Marcelo Pereyra. "Wasserstein control of mirror langevin monte carlo". In: *Conference on Learning Theory*. PMLR. 2020, pp. 3814–3841.

[Zha06] Fuzhen Zhang. *The Schur complement and its applications*. Vol. 4. Springer Science & Business Media, 2006.

[Zho02] Ding-Xuan Zhou. "The covering number in learning theory". In: *Journal of Complexity* 18.3 (2002), pp. 739–767.

[ZHP19] Yan Zhang, Jonathon Hare, and Adam Prugel-Bennett. "Deep set prediction networks". In: *Advances in Neural Information Processing Systems* 32 (2019), pp. 3212–3222.

[ZLM21] Yichao Zhou, Shichen Liu, and Yi Ma. "NeRD: Neural 3D Reflection Symmetry Detector". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 15940–15949.

[ZSS22] Paul Zhang, Dmitriy Smirnov, and Justin Solomon. "Wassersplines for Neural Vector Field-Controlled Animation". In: *Computer Graphics Forum*. Vol. 41. 8. Wiley Online Library. 2022, pp. 31–41.