Near-Optimal Herding

by

Samira Samadi

B.Sc., Sharif University of Technology, 2012

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

in

The Faculty of Graduate and Postdoctoral Studies

(Computer Science)

THE UNIVERSITY OF BRITISH COLUMBIA

(Vancouver)

August 2014

Abstract

Herding is an algorithm of recent interest in the machine learning community, motivated by inference in Markov random fields. It solves the following Sampling Problem: given a set $\mathcal{X} \subset \mathbb{R}^d$ with mean μ, construct an infinite sequence of points from $\mathcal{X}$ such that, for every $t \geq 1$, the mean of the first t points in that sequence lies within Euclidean distance $O(1/t)$ of μ. The error of a solution to Sampling Problem is defined to be the distance between the empirical mean of the first t samples and the original mean μ.

The $O(1/t)$ error bound suppresses the dependence on d and $\mathcal{X}$. In this thesis, we study the best dependence on d and $|\mathcal{X}|$ that can be achieved for the error in Sampling Problem. Known analysis of the Herding algorithm give an error bound that depends on geometric properties of $\mathcal{X}$ but, even under favorable conditions, this bound depends linearly on d.

We first show that any algorithm for the Sampling Problem must have error $\Omega(\sqrt{d}/t)$. Afterward, we present a new polynomial-time algorithm that solves the Sampling Problem with error $O\big(\sqrt{d}\log^{2.5}|\mathcal{X}|/t\big)$ assuming that $\mathcal{X}$ is finite. This implies that our algorithm is optimal to within logarithmic factors. Finally, we prove that the actual error of the Herding algorithm is strictly worse than the error of our algorithm if we measure the error in the ℓ_∞-norm. Our algorithm is randomized and based on recent algorithmic results in discrepancy theory. We implement our algorithm and other potential solutions for the Sampling Problem and evaluate them on various inputs.

Preface

This work is the product of great deal of active discussions between me and my supervisor Prof. Nick Harvey and numerous refinements. The theoretical contributions is largely based on the work that has been published at *Conference on Learning Thoery 2014* [21]. All the algorithms are implemented in Matlab and the plots are added for the experimental evaluations.

Table of Contents

List of Figures

Acknowledgements

First and foremost, I would like to express my utmost gratitude to my supervisor, Prof. Nick Harvey. He has devoted countless hours to myself and my work and provided me insightful guidance and endless support. I immensely enjoyed and learned from every meeting I had with him and I am deeply indebted to him for being a great mentor. I would like to extend my sincerest thanks to Prof. Alexandre Bouchard-Côté for being my second reader and for his helpful comments. I would like to thank Prof. David Kirkpatrick for his continues support and valuable feedback.

I also owe my special thanks to my friends and colleagues in UBC, too numerous to name here, for their help and for making my life much more enjoyable. I would very much like to thank Maryam Siahbani and Shabnam Mirshokraie for their true kindness and support during my stay in Vancouver.

Last but not least, I would like to thank my family for their tireless support, love and trust. None of this would have been possible without their love and patience.

Dedication

To my brother, Mehdi, for always being there for me.

Chapter 1

Introduction

Herding is a topic that has attracted much interest in the machine learning community over the past few years. It was originally proposed [33] as a method for "sampling" from a Markov random field[1] that agrees with a given data set $\mathcal{X}$. The traditional approach for this scenario is to use maximum likelihood estimation to infer parameters of the model, and once the model is obtained, one can draw samples from it.

A drawback of the traditional approach is that the maximum likelihood estimation step can be computationally intractable. Another difficulty is that the estimated model can have different local modes and it may cause the algorithm to get stuck in one of the local modes. The Herding approach sidesteps these intractability problems; it does not infer the model parameters, but instead deterministically produces a sequence of "samples" whose moments rapidly converge to the moments of the given data set $\mathcal{X}$.

A concise statement of the Herding algorithm is as follows

$$\begin{aligned} x_{t+1} &\in \arg\max_{x \in \mathcal{X}} \langle\, w_t,\, x \,\rangle \\ w_{t+1} &= w_t + \mu - x_{t+1} \end{aligned}$$

where x_t is the sample that the algorithm produces in step t and w_t is a *weight vector* that guides the choice of the next sample. The initial weight vector w_0 is initialized arbitrary. The error of a sampling algorithm after t steps, is defined as the Euclidean distance between the empirical mean of the first t samples and the original mean μ. The convergence rate of a sampling algorithm is the error at each step as a function of t.

The Herding algorithm has several attractive properties. Unlike previous sampling techniques that used random number generation, Herding is fully deterministic. Computationally, it is very simple; it is a greedy algorithm whose pseudocode is only two lines. Yet despite its simplicity, it has a dignified lineage. Herding is related to to conditional gradient methods [4] and to quadrature methods [4, 23].

[1] Markov random field is a synonym for undirected graphical model.

There is also a deep connection between the Herding algorithm and the Perceptron method [19]. It was observed that supervised Perceptron algorithm and the Herding algorithm can both be derived from the classic "Perceptron Cycling Theorem" [26]. We will discuss this connection in more details in Section 2.5.

Herding was originally proposed for discrete space processes but it can naturally be extended to continuous spaces using the kernel trick [17]. The Kernel trick is a method for implicitly mapping the observations coming from a general set $\mathcal{X}$ into a high-dimensional inner product space $\mathcal{V}$ such that the observations get well-behaved linear structure in $\mathcal{V}$. The resulting kernel Herding algorithm can be applied to accommodate a large numbers of features.

Finally, an intriguing property is that the samples produced by Herding converge *faster* than independent random samples, in the sense that the error after t Herding samples is inversely proportional to t, whereas the error of t random samples is inversely proportional to $\sqrt{t}$ [17] [2]. The Herding algorithm keeps track of all the samples generated so far and avoids sampling from the areas that have already been over sampled. This helps the algorithm to have fast convergence rate.

The purpose of this thesis is to rigorously analyze the theoretical underpinnings of the Herding algorithm. In particular, since Herding's convergence properties have received much attention, we wish to determine the best convergence achievable by *any* algorithm. For this purpose, we first need to define a mathematical setting for our target class of sampling algorithms.

1.1 Our Contribution

We define an exact mathematical problem that asks "How can we perform an efficient sampling?" We call this problem the *Sampling Problem*. Our goal is to analyze any possible algorithm that gives a solution to this problem.

One of our main observations is that there is a connection between the Sampling Problem and *discrepancy theory*, an important topic in combinatorics and geometry. We exploit this connection by restating the Sampling Problem in the geometrical setting established by Banaszczyk [8]. The new statement of the problem allows us to find a close relation between the Sampling Problem and a well-studied problem in discrepancy theory which we call the *Permutation Problem*. We show that there is an algorithmic reduction from the Permutation Problem to the Sampling Problem and focus on solving the Permutation Problem.

[2] The proof given in [17] for the error of Herding algorithm seems not to be precise, we will give a correct version of it in Chapter 3.

To begin, we present a fairly simple lower bound on the convergence rate of any algorithm for the Sampling Problem (see Section 5.5). We then observe that this lower bound is actually optimal if a classic conjecture in discrepancy theory is true. Next, we present a partial result towards this conjecture which is algorithmic and uses recent breakthroughs in discrepancy theory [9, 24]. This yields a new, polynomial-time algorithm to produce a sequence of samples whose convergence rate is within a logarithmic factor of optimal. In contrast, the best known upper bound on the convergence of the Herding algorithm is significantly worse, even under generous assumptions. We conjecture that the actual convergence of the Herding algorithm is strictly worse than the convergence of our new algorithm, and we prove that this is true if convergence is measured in the ℓ_∞-norm.

1.2 Related Work

The invention of the Herding algorithm goes back to 2009 when Welling [33] first proposed it as a new method for directly generating pseudosamples from the observed moments of a model. One can formulate this problem as the general *maximum entropy problem* [32] as follows. We are given the pairwise marginal distributions P_{ij} and we would like to estimate the joint distribution $P(x)$ that has the maximum entropy. Once we have estimated the full joint distribution, we can draw samples from it. The learning phase in the latter formulation is computationally expensive and this is what makes the Herding algorithm an attractive approach; Herding avoids this difficulty by directly generating samples.

Although Welling [33] did not focus on the theoretical foundation of the Herding algorithm, later on the researchers tied this algorithm to the well-studied areas in learning theory and continues optimization. Chen et al. [17] consider Herding as an infinite memory process and explain it as a mapping $x_{t+1} = G(x_1, \ldots, x_t, w_0)$. In this scenario, the Herding algorithm can be interpreted as taking gradient steps of size 1 on function G. Another observation is that it is easy to apply the kernel trick to the Herding algorithm. This is due to the fact that in the Herding algorithm, the formula for obtaining the next sample x_{t+1} only deals with inner product. This results in a kernel Herding algorithm and its error function that decreases as a function of $O(1/t)$.

Another line of work [4] showed that the Herding algorithm is related to the *conditional gradient algorithm* also called as the *Frank-Wolfe* algorithm. Frank-Wolfe algorithms are a class of iterative optimization algorithms that has applications in many areas including machine learning, signal processing and statistics. These algorithms can be applied to any smooth convex function with a compact

convex domain. An attractive property of Frank-Wolfe algorithms is that they do not perform expensive operations. At each iteration, they only need to compute the gradient of the objective function and optimize linear functions on its domain. This is unlike other optimization methods such as "gradient descent" algorithm that requires a projection step at each iteration. The connection between Herding and Frank-Wolfe allows the authors to suggest better approaches for the task of approximation integrals in a reproducing kernel Hilbert space.

From another perspective, Herding is related to deterministic algorithms for sampling from distributions [16, 17, 22, 23]. For many researchers, it is very interesting to pursue Herding from this line of work, as it provides a well defines mathematical setting for it. In particular, since Herding convergence rate has attracted much attention, recent work has studied using Herding ideas to improve convergence of the Gibbs sampler [14], this relates to interesting directions in combinatorics [22] and Markov chain Monte Carlo methods [16].

1.3 Thesis Organization

The rest of this thesis is organized as follows. In Chapter 2, we present formal definitions and briefly explain our main results. In Chapter 3, we theoretically prove some of the important existing results. In Chapter 4, we present a brief overview of discrepancy theory which is necessary to explain our results in detail. In Chapter 5, we present our main algorithm and analyze its convergence rate. In Chapter 6, we conclude our work and discuss future directions.

Chapter 2

Formal Definitions and Main Results

In this chapter we will define the Sampling Problem which is the main problem of study in this thesis. We continue by discussing various solutions to the Sampling Problem including Random sampling and the Herding algorithm. At each section, we provide theoretical results on the upper bound analysis for the error of these algorithms as well as experimental evaluations of their performance on different data sets[3]. For now, we do not go into the proofs of the theoretical results but we will explain all the details in Chapter 3. The goal of this chapter is to give the reader a good understanding of the Sampling Problem and properties of its current solutions.

The following notation will be used throughout the thesis. We write $[n]$ for the set $\{1, 2, \ldots, n\}$. For any matrix M we let $M_{i,*}$ refer to the i^{th} row of M, and $M_{*,j}$ refer to the j^{th} column of M. All logarithms are in base 2. The ℓ_p-norm is denoted $\|\cdot\|_p$. The i^{th} standard basis vector is denoted e_i.

2.1 Sampling Problem

Let $\mathcal{X} \subset \mathbb{R}^d$ be a finite set with $|\mathcal{X}| = n$. Let $\mu = \sum_{x \in \mathcal{X}} x/n$ and $\delta = \max_{x \in \mathcal{X}} \|x\|_2$. Let r be the maximum value such that the ℓ_2-ball centered at μ of radius r is contained in the convex hull of $\mathcal{X}$ (See figure 2.1).

More generally, one could assume that $\mathcal{X}$ is infinite and that it lies in an infinite-dimensional Hilbert space. For now we will restrict to the case of finite $\mathcal{X}$ and finite dimensions but in Section 6.2 we will briefly talk about the infinite case too.

[3] All the functions are implemented in Matlab.

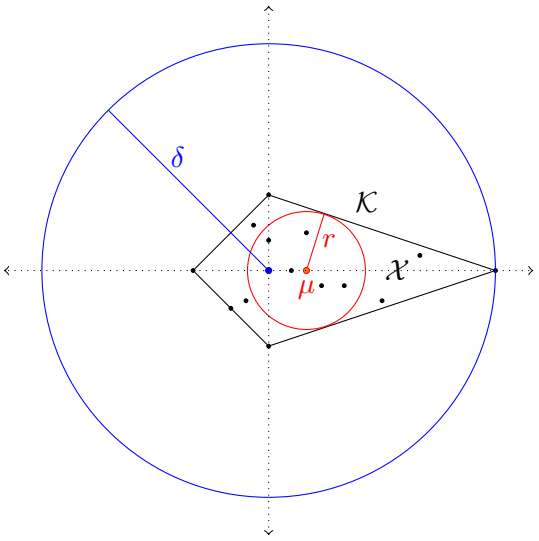

Figure 2.1: The figure shows: Set of points $\mathcal{X}$ and its convex hull $\mathcal{K}$ (black). The smallest circle centred at the origin that contains $\mathcal{K}$ and its radius δ (blue). The biggest circle centred at μ that is contained in $\mathcal{K}$ and its radius r (red).

Definition 2.1.1 (Sampling Problem). We wish to find an infinite sequence of points $x_1, x_2, \ldots$ from the set $\mathcal{X}$. Each x_i is called a *pseudosample*. The *error* of the first t pseudosamples is $\|\sum_{i\leq t} x_i/t - \mu\|_2$. The goal is to ensure that, for all $t \geq 1$, the error of the first t pseudosamples is at most α/t, where α is a quantity can depend arbitrarily on d, n and $\mathcal{X}$ but it should not depend on t.

Our goal is to rigorously analyze the Sampling Problem and to find an algorithm that solves this problem with the best possible error. We will start by considering Random sampling and see if this algorithm gives a solution to Sampling Problem.

2.2 Random Sampling

The most natural algorithm for solving Sampling Problem is Random sampling. The formal definition of this algorithm can be states as following:

Definition 2.2.1 (Random Sampling). Get $i.i.d.$ uniform samples from the set $\mathcal{X}$.

We evaluate the error function for Random sampling both theoretically and experimentally. Standard analysis for Random sampling shows that the error for this

algorithm decreases as a function of $O(1/\sqrt{t})$[4]. This proves that Random sampling does not solve the Sampling Problem. In order to understand the performance of this algorithm in the experiments, we first need to introduce the data sets that we are going to use throughout this chapter.

2.3 Data Sets

We show any finite data set $\mathcal{X} \in \mathbb{R}^d$ (with $|\mathcal{X}| = n$) as columns of a $d \times n$ matrix. Here we introduce four data sets that we will use in our experiments. For all the matrixes, we add a normalization step to make $\mu = 0$ and $\delta = 1$.

(1) Consider the random matrix $\mathcal{F}_{d\times n}$ such that each entry of $\mathcal{F}$ is drawn independently from the standard normal distribution.

(2) Let $\mathcal{G}_{d\times n}$ be the identity matrix.

(3) Let matrix $\mathcal{H}_{d\times n}$ be defined explicitly as follows:

$$\mathcal{H}_{d\times n} = \begin{bmatrix} \overbrace{1/\sqrt{d}}^{\text{A}} & \overbrace{-\epsilon & \cdots & -\epsilon}^{\text{B}} & \overbrace{0 & \cdots & 0}^{\text{C}} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 1/\sqrt{d} & -\epsilon & \cdots & -\epsilon & 0 & \cdots & 0 \\ 1 & 0 & \cdots & 0 & -\epsilon & \cdots & -\epsilon \end{bmatrix}.$$

where $|A| = 1$, $|B| = \left\lceil \sqrt{d} \right\rceil$, $|C| = d$ and $\epsilon = 1/d$. This gives us $n = d + \left\lceil \sqrt{d} \right\rceil + 1$.

Figure 2.2 shows the performance of Random sampling when applied to matrices $\mathcal{F}, \mathcal{G}$ and $\mathcal{H}$ with $d = 20, n = 25$ to generate 50 pseudosamples.

[4] We prove this result in Section 3.1.

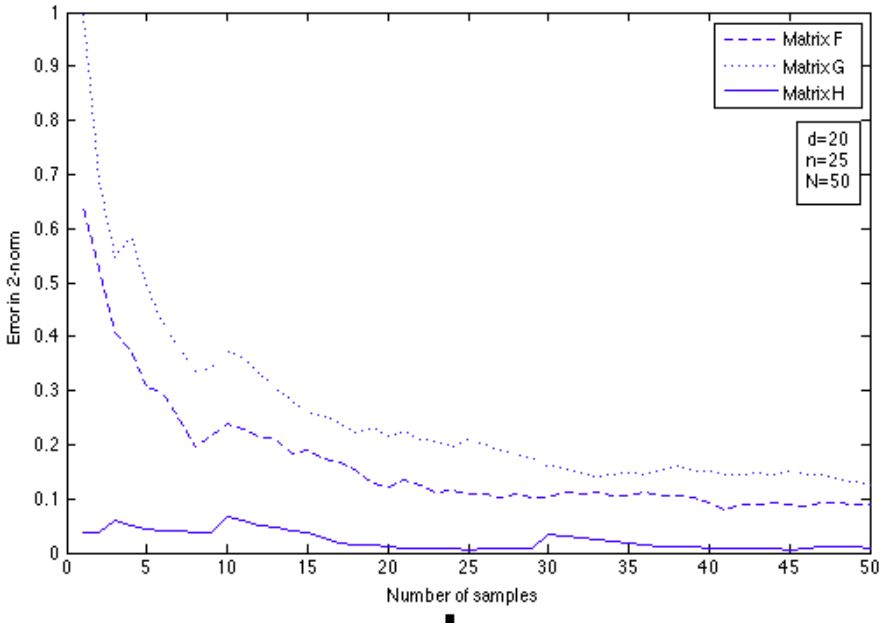

Figure 2.2: Random sampling performance on different inputs.

So far we have presented Random sampling and seen that this algorithm does not give a solution to Sampling Problem. The next natural idea for solving Sampling Problem is a greedy algorithm which we will explain in the next section.

2.4 Herding Algorithm

Herding algorithm was first introduced by Welling [33] as method for learning in intractable Markov random fields models. It is a deterministic algorithm that directly generates samples from the model of study. A concise statement of the Herding algorithm is as follows [5]

$$x_{t+1} \in \arg\max_{x \in \mathcal{X}} \langle w_t, x \rangle \tag{2.4.1a}$$

$$w_{t+1} = w_t + \mu - x_{t+1} \tag{2.4.1b}$$

The initial weight vector w_0 is initialized somewhat arbitrarily. The literature

[5] In the definition of Herding algorithm, we will also assume that the algorithm choses the smallest index in the set $\{\arg\max_{x \in \mathcal{X}} \langle w_t, x \rangle\}$. This assumption is used in the proof of Theorem 2.7.3.

has proposed setting $w_0 = 0$ or $w_0 = \mu$ [4, 17]. In Section 3.2 we show that the error function of Herding algorithm is $O\big((\|w_0\|_2 + \delta^2/r)/t\big)$.

Figure 2.3 shows the performance of Herding algorithm when applied to matrices $\mathcal{F}, \mathcal{G}$ and $\mathcal{H}$ with $d = 20$ and $n = d + \sqrt{d} + 1 = 25$. The algorithm starts with $w_0 = 0$ and generates 50 pseudosamples. The top plot shows the error in $\|\cdot\|_2$ and the bottom plot in $\|\cdot\|_\infty$. Comparing this figure with the Figure 2.2, it is easy to see that the 2-norm error of Herding is much better than Random sampling.

In Section 5.6 we will show that the error of Herding algorithm is strictly worse than the error of our algorithm if we compute the error in the ℓ_∞-norm. This is why we include the ℓ_∞-norm evaluation of the error for Herding algorithm in the Figure 2.3.

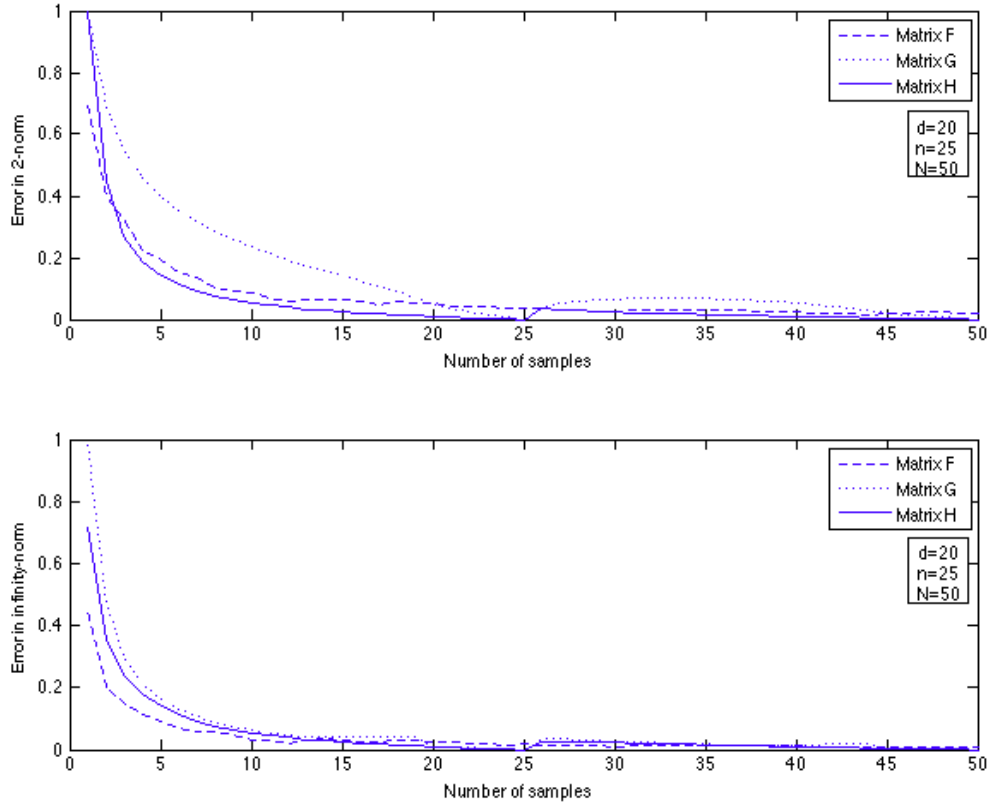

Figure 2.3: Herding algorithm performance on different inputs.

2.4.1 The choice of w_0 is important

In the previous section, we analyzed the performance of the Herding algorithm when it starts with $w_0 = \mu = 0$. It is interesting to see that choice of the initial weight vector w_0 plays an important role in the analysis of the error for Herding algorithm.

We run the Herding algorithm on matrix $\mathcal{H}$ to generate 150 pseudosamples. Figure 2.4 shows the $\|\cdot\|_2$ error of this algorithm if we start with different weight vectors w_0.

- Dashed line shows error of Herding algorithm for $w_0 = (0, \cdots, 0)^\mathsf{T}$.
- Solid line shows error of Herding algorithm for $w_0 = (1, \cdots, 1)^\mathsf{T}$.
- Dotted line shows error of Herding algorithm for $w_0 = (d, \cdots, d)^\mathsf{T}$.

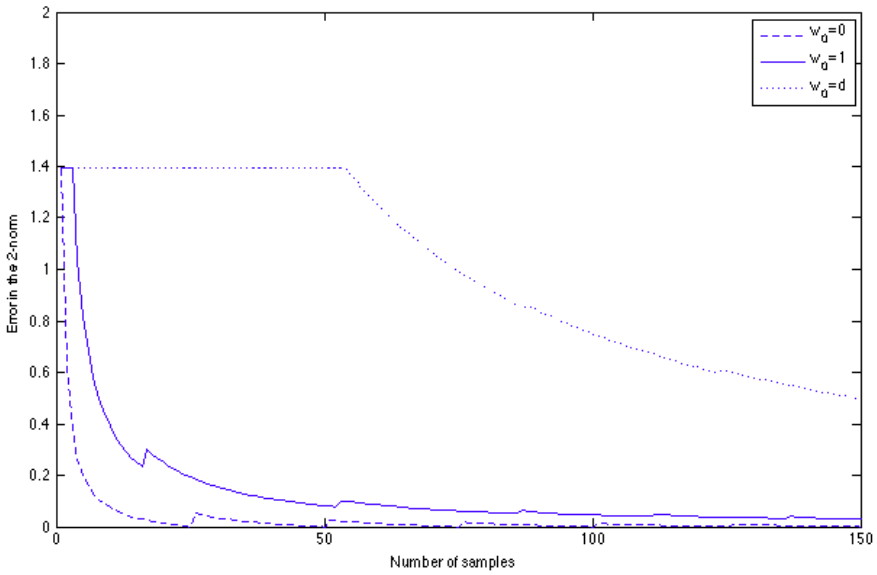

Figure 2.4: Herding algorithm error depends on w_0.

Our observation is that there is a direct connection between the error for the Herding algorithm and the $\|w_0\|_2$.

2.5 Perceptron

So far we have talked about Random sampling and the Herding algorithm as potential solutions for the Sampling Problem. We observed that the Herding algorithm solves the Sampling Problem with error that decreases as a function of $O\big((\|w_0\|_2 + \delta^2/r)/t\big)$. Initially it may seem intriguing that the error is proportional to $1/t$, but this is an instance of a much more general phenomenon which we will talk about in this section.

The Perceptron algorithm was first introduced by Rosenblatt as a method for learning relations in brain data [18]. Later on, it was implemented as "Mark 1 Perceptron" hardware and was used as an AI machine for image recognition. At the beginning of its invention, the Perception seemed to be having a big influence in the AI community. However, after a few years it was proved that it can not be applied to a big class of data sets. In this section, we introduce the Perceptron as an online algorithm for learning a linear threshold function.

We present the Perceptron algorithm in the *online learning* setting, which is defined as follows [5]. The algorithm receives an unlabelled example and predicts a classification of this example using its current hypothesis. In the next step, the correct answer is presented to the algorithm and the process repeats.

Assume that we have a set of examples $\{x_1, \ldots, x_n\}$ with the labels $\{y_1, \ldots, y_n\} \in \{\pm 1\}^n$. The goal of the Perceptron algorithm is to learn a linear classifier

$$w_1 x_1 + w_2 x_2 + \ldots + w_n x_n > 0$$

with the following procedure [5]:

Algorithm 1 Perceptron

Start: Set $t = 1$ and $w_1 = 0$

Iterative Step:

- Given example x_t, predict positive iff $w_t \cdot x_t > 0$
- For wrong prediction, update w_{t+1} as follows:
  - Mistake on positive: $w_{t+1} = w_t + x$
  - Mistake on negative: $w_{t+1} = w_t - x$
- $t = t + 1$

There is a connection between the classic Perceptron algorithm and the Herding algorithm; the general "Perceptron Cycling Theorem" applies to both of them. It

was observed [19] that if Equation (2.4.1a)

$$x_{t+1} \in \arg\max_{x \in \mathcal{X}} \langle w_t, x \rangle$$

is replaced with any rule that ensures $\langle w_t, x_{t+1} \rangle \geq 0$, then the Perceptron Cycling Theorem [13] implies that the error is at most α/t for some quantity α. However, the proof of this theorem uses compactness and does not give any quantitative bound on α. Giving an effective bound on α was an open question for decades, until [3] recently proved an explicit bound that is exponential in d.

2.6 Towards Optimal Bound for Sampling Problem

As we have seen, the Perceptron Cycling Theorem implies that the Herding algorithm solves the Sampling Problem with error function that is proved to be bounded by α/t for some quantity α. This shows that the interesting aspect of the Sampling Problem is not that the error is proportional to $1/t$, but rather determining a precise value of α such that the error is at most α/t. The purpose of this thesis is to address the following question, which seems fundamental.

Question 2.6.1. What is the best value of α (as a function of d, n or $\mathcal{X}$) that can be obtained in the Sampling Problem?

The analysis of the Herding algorithm gives a bound on α that depends on r, and hence on the geometry of $\mathcal{X}$. An undesirable aspect of this bound is that it does not reflect the *intrinsic* geometry: it is highly sensitive to rescalings of $\mathcal{X}$. Imagine multiplying the first coordinate of all points in $\mathcal{X}$ by a quantity ϵ and keeping other coordinates unchanged — if the coordinates represent features of the data, then this amounts to simply changing the units of the first feature. One can see that, as $\epsilon \to 0$, r decreases by a factor of roughly ϵ, whereas δ remains mostly unchanged. Thus, such scalings can arbitrarily increase the ratio δ/r, and hence the error bound of the Herding algorithm. This motivates the need for a bound on α that is not disrupted by unimportant rescalings. In the next section, we introduce a geometrical setting that fulfills these conditions.

2.6.1 John's position

For any convex body $\mathcal{K}$, there is a canonical transformation that negates the effects of such unimportant rescalings. We say that $\mathcal{K}$ is in *John's position* if the largest ellipsoid contained in $\mathcal{K}$ is a scalar multiple of the unit Euclidean ball (see [6, 30]). So consider $\mathcal{K}$ to be the convex hull of $\mathcal{X}$, and assume that the center of this ball

is μ. Then the quantity r defined above is the radius of this ball. John's theorem asserts that $\mathcal{K}$ is contained in the ball centered at μ of radius rd. Assuming that $\mu = 0$, it follows that the quantity δ defined above is at most rd, and hence[6] $\delta/r \leq d$. Thus, under these assumptions, the error of the Herding algorithm is $O\big((\|w_0\|_2 + \delta d)/t\big)$.

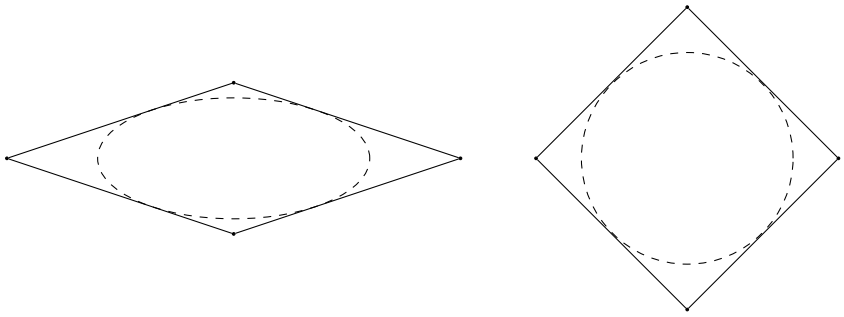

Figure 2.5: The body in the left is not in John's position but the body in the right is in John's position.

An attractive aspect of this bound is that it essentially depends only on d, which is a fundamental parameter of the Sampling Problem. This suggests two natural questions:

- ◇ First, does a bound depending only on d hold without making the assumptions of this section?
- ◇ Second, is this linear dependence on d optimal?

We investigate the answers to this questions in the next section.

2.7 Main Results

In this section, we present the main contribution of this thesis toward answering the above questions. Our first, and rather simple, observation is:

Theorem 2.7.1. For each d, there exists a set $\mathcal{X} \subset \mathbb{R}^d$ with $\max_{x\in\mathcal{X}} \|x\|_2 = 1$ such that every solution to the Sampling Problem on $\mathcal{X}$ has error $\Omega(\sqrt{d}/t)$ for $t = \Theta(d^2)$.

[6] This bound is tight: a simplex centered at the origin has δ/r exactly equal to d.

If we allow error bounds that depend on n, then we can show a nearly-matching upper bound.

Theorem 2.7.2. There is an algorithm with running time $\mathrm{poly}(d, n)$ that solves the Sampling Problem with error $O(\sqrt{d}\delta \log^{2.5}(n)/t)$.

Theorem 2.7.1 is proven in Section 5.5 and Theorem 2.7.2 is proven in Section 5.3. We do not know whether the algorithm of Theorem 2.7.2 has strictly better error than the Herding algorithm. However, if we measure error in the ℓ_∞-norm then our algorithm is seen to be strictly better, assuming that $n = 2^{o(d^{1/5})}$.

Theorem 2.7.3. For each d, we can construct a set $\mathcal{X} \in \mathbb{R}^d$ such that the Herding algorithm has $\max_t \|t \cdot (\sum_{i=1}^t x_i/t - \mu)\|_\infty = \Omega(\sqrt{d})$. On the other hand, the algorithm of Theorem 2.7.2 has $\max_t \|t \cdot (\sum_{i=1}^t x_i/t - \mu)\|_\infty = O(\log^{2.5}(n))$.

Chapter 3

Analysis of the Existing Results

In this chapter we formally prove the upper bound results that we earlier mentioned for Random sampling and the Herding algorithm. In Section 3.1, we prove that the error of Random sampling after generating t samples is a function of $O(1/\sqrt{t})$. Furthermore, it can be seen that this bound is tight which proves that Random sampling does not give a solution to the Sampling Problem. In Section 3.2, we prove that the error of the Herding algorithm after generating t samples decreases as a function of $O((\|w_0\|_2 + \delta^2/r)/t)$.

3.1 Random Sampling Error Analysis

We prove that the error of Random sampling on the set $\mathcal{X}$ is $O(1/\sqrt{t})$. For the sake of this thesis, we are not concerned about other factors suppressed by the O notation in $O(1/\sqrt{t})$ as the $1/\sqrt{t}$ dependence is enough to show that Random sampling does not solve Sampling Problem.

We are given the set $\mathcal{X}$ with assumptions of Section 2.1. For sake of simplicity, we also assume that $0 \leq x \leq 1$ for all $x \in \mathcal{X}$ [7]. Let $X_1, X_2, \ldots, X_t, \ldots$ be the random variables that represent pseudosamples generated by Random sampling. For every $j \in [t]$, we can write $X_j = (X_{1j}, \ldots, X_{dj})^\mathsf{T}$. Our approach is to apply the Chernoff bound to bound tail of the distribution for random variable $\sum_{j=1}^{t} X_{ij}$ for all $i \in [d]$. Afterward, we can use union bound and norm inequalities to give a bound on $\left\| \sum_{j=1}^{t} X_j/t - \mu \right\|_2$.

The Chernoff bound is one of the most powerful tools in probability theory. It was first introduced by Herman Rubin [31] to show that sum of independent random variables has a distribution for which the tail is bounded by an exponentially decreasing function. One statement of Chernoff bound can be stated as follows [20, 27]:

Theorem 3.1.1 (Chernoff Upper Tail). Let $Z_1, \ldots, Z_m$ be independent random

[7] The proof can be generalized for any bounded interval.

variables such that Z_i always lies in the interval $[0, 1]$. Let $Z = Z_1 + \ldots + Z_m$ and $\mu = \mathbb{E}[Z]$. Let $p_i = \mathbb{E}[Z_i]$ and note that $\mu = \sum_i p_i$. Then for any $\delta > 0$

$$\Pr[Z \geq (1+\delta)\mu] \leq \exp\big(-\mu((1+\delta)\ln(1+\delta) - \delta)\big)$$

For any $\delta \in [0, 1]$

$$\Pr[Z \geq (1+\delta)\mu] \leq \exp(-\mu\delta^2/3) \tag{3.1.1}$$

For any $\delta \geq 1$

$$\Pr[Z \geq (1+\delta)\mu] \leq \exp(-\mu\delta/3) \tag{3.1.2}$$

Consider the sequence of independent random variables $X_{i1}, X_{i2}, \ldots, X_{it}$ such that $i \in [d]$. We define a new random variable $\sum_{j=1}^{t} X_{ij}$. Note that $\mathbb{E}[\sum_{j=1}^{t} X_{ij}] = t\mathbb{E}[X_{i1}] = t\mu_i$. Now we can apply Chernoff bound to random variable $\sum_{j=1}^{t} X_{ij}$ with $\delta = \frac{30 \times \ln d}{\sqrt{t}\mu_i}$. From Equation 3.1.1 we know that for $\delta \in [0, 1]$

$$\Pr[\sum_{j=1}^{t} X_{ij} \geq (1+\delta)t\mu_i] \leq \exp(-\frac{300\ln^2 d}{\mu_i}) \leq \exp(-300\ln^2 d)$$

This is due to the fact that $\mu_i \leq 1$. From Equation 3.1.2, for $\delta \geq 1$

$$\Pr[\sum_{j=1}^{t} X_{ij} \geq (1+\delta)t\mu_i] \leq \exp(-10\ln d\sqrt{t}) \leq \exp(-10\ln d)$$

Thus, for any $\delta \geq 0$

$$\Pr[\sum_{j=1}^{t} X_{ij}/t - \mu_i \geq \delta\mu_i] = \Pr[\sum_{j=1}^{t} X_{ij} \leq (1+\delta)t\mu_i] \leq \exp(-10\ln d)$$

Since $\delta\mu_i = \frac{30\ln d}{\sqrt{t}}$

$$\Pr[\sum_{j=1}^{t} X_{ij}/t - \mu_i \geq \frac{30\ln d}{\sqrt{t}}] \leq \exp(-10\ln d) = d^{-10} \tag{3.1.3}$$

Let's define event A_i to be the event of $\sum_{j=1}^{t} X_{ij}/t - \mu_i \leq \frac{30\ln d}{\sqrt{t}}$. We want to compute $\Pr[\bigcap_{i=1}^{d} A_i]$. Using union bound and Equation 3.1.3

$$\Pr[\bigcap_{i=1}^{d} A_i] = 1 - \Pr[\bigcup_{i=1}^{d} A_i^{\complement}] \geq 1 - (\sum_{i=1}^{d} \Pr[A_i^{\complement}]) \geq 1 - d^{-9}$$

Thus, with constant probability $1 - d^{-9}$ for all $i \in [d]$

$$\sum_{j=1}^{t} X_{ij}/t - \mu_i \leq \frac{30 \ln d}{\sqrt{t}}$$

So with constant probability $1 - d^{-9}$ we have that

$$\left\| \sum_{j=1}^{t} X_j/t - \mu \right\|_2 = \sqrt{\sum_{i=1}^{d} (\sum_{j=1}^{t} X_{ij}/t - \mu_i)^2} \leq \frac{30\sqrt{d} \ln d}{\sqrt{t}}$$

This shows that the error of t pseudosamples generated by Random sampling is $O(\sqrt{d} \ln d/\sqrt{t})$. This bound can be improved as a function of d if we make better choices of δ. One can complete the proof by showing that this bound is tight.

3.2 Herding Error Analysis

In Section 2.1, we introduced the Herding algorithm and showed its performance on different data sets. Remember that this algorithm is defined as

$$\begin{aligned} x_{t+1} &\in \arg\max_{x \in \mathcal{X}} \langle\, w_t,\, x \,\rangle \\ w_{t+1} &= w_t + \mu - x_{t+1} \end{aligned}$$

In this section we theoretically analyze the error for Herding algorithm and prove the upper bound $O((\|w_0\|_2 + \delta^2/r)/t)$. The argument that we give here is a modification of the proof in [17].

Assume that the Herding algorithm generates pseudosamples $x_1, x_2, \ldots, x_t, \ldots$ and weights $w_0, w_1, w_2, \ldots, w_t, \ldots$. We have that $w_i = w_{i-1} + \mu - x_i$ for all $1 \leq i \leq t$. Summing these above equations we get that $w_t = w_0 + t\mu - \sum_{i=1}^{t} x_i$. The error of the Herding algorithm for the first t samples is

$$\left\| \frac{1}{t} \sum_{i=1}^{t} x_i - \mu \right\|_2 = \frac{1}{t} \left\| \sum_{i=1}^{t} x_i - t\mu \right\|_2 = \frac{1}{t} \|w_0 - w_t\|_2 \leq \frac{1}{t} (\|w_0\|_2 + \|w_t\|_2)$$

So in order to bound the error, we only need to bound $\|w_t\|_2$. We claim that $\|w_t\|_2 \leq O(\delta^2/r)$. Let's define the convex body $\mathcal{C} = \mathcal{K} - \mu$. Since $\|x\|_2 \leq \delta$ for all $x \in \mathcal{K}$, $\|c\|_2 \leq 2\delta$ for all $c \in \mathcal{C}$. Let $c_t = \arg\max_{c \in \mathcal{C}} \langle w_t, c \rangle$ for all t. The Herding algorithm second equation becomes $w_{t+1} = w_t - c_t$. Since there is a ball

of radius r with centre μ inside $\mathcal{K}$ and $\mathcal{C}$ is a shifting of $\mathcal{K}$, there is a ball of radius r with centre 0 inside $\mathcal{C}$. Thus $r \cdot \frac{w_t}{\|w_t\|_2} \in \mathcal{C}$.

Consider the sequence $w_0, w_1, \ldots$ and let k be the first index for which $\|w_k\|_2 > 2\delta^2/r$. Then

$$\|w_{k+1}\|_2^2 - \|w_k\|_2^2 = \|c_k\|_2^2 - 2\langle w_k, c_k\rangle \leq 4\delta^2 - 2\langle w_k, r \cdot \frac{w_k}{\|w_k\|_2}\rangle = 4\delta^2 - 2r\,\|w_k\|_2 < 0.$$

Therefore $\|w_k\|_2 > \|w_{k+1}\|_2$. This means that as soon as the norm of a weight vector gets bigger than $2\delta^2/r$, the norm starts decreasing until it gets smaller than $\frac{2\delta^2}{r}$ again. Since $w_k = w_{k-1} - c_{k-1}$, we have

$$\|w_k\|_2 \leq \|w_{k-1}\|_2 + \|c_{k-1}\|_2 \leq \frac{2\delta^2}{r} + 2\delta = O(\frac{\delta^2}{r})$$

Note that this is the maximum value that the norm of any vector w_t can get. Therefore the error of Herding algorithm is bounded by

$$\left\| \frac{1}{t} \sum_{i=1}^{t} x_i - \mu \right\|_2 \leq \frac{1}{t}(\|w_0\|_2 + \|w_t\|_2) \leq O(\|w_0\|_2 + \frac{\delta^2}{r})/t.$$

Chapter 4

Discrepancy Theory

Discrepancy theory is a major area of study in both combinatorics and geometry [15, 25]. The powerful results in this area have applications in many theoretical areas of computer science. We begin this section by defining a stringent problem that is very similar to the Sampling Problem.

4.1 Permutation Problem

Definition 4.1.1 (Permutation Problem). Let $\mathcal{X}$, μ, d and n be as before. We wish to find a bijection $\pi : \{1, \ldots, n\} \to \mathcal{X}$. The *error* of the first t points is $\|\sum_{i \leq t} \pi(i)/t - \mu\|_2$. The goal is to ensure that, for all $t \geq 1$, the error of the first t points is at most α/t.

It is easy to see that any solution to the Permutation Problem yields a solution to the Sampling Problem with the same parameter α. To see this, consider the infinite sequence obtained by concatenating copies of $\mathcal{X}$, each ordered by π. Every contiguous subsequence of length n sums to $n\mu$. So, for every length-t prefix of this infinite sequence, only the last $t \bmod n$ vectors contribute to the error. The error of those last vectors is exactly $\|\sum_{i=1}^{t \bmod n}(\pi(i) - \mu)/t\|_2$, which is assumed to be at most α/t.

Thus, to give an upper bound on the error in the Sampling Problem, it suffices to give an upper bound on the error in the Permutation Problem. This is the approach used to prove Theorem 2.7.2. To explain this approach in more detail, we must introduce tools from discrepancy theory.

Fix a convex body $B \subset \mathbb{R}^d$ that is 0-symmetric (i.e., $B = -B$). For any finite set $V \subset B$ with $\sum_{v \in V} v = 0$, let β_V be the smallest value for which there is an ordering $v_1, v_2, \ldots$ of V with $\sum_{i \leq t} v_i \in \beta_V \cdot B$ for all t. The *Steinitz constant* of B, denoted $S(B)$, is the supremum of β_V over all finite $V \subset B$. Note that $S(B)$ depends only on d and B, and is not a function of $|V|$.

The connection between the Permutation Problem and discrepancy theory is now apparent. Let B_2^d be the Euclidean unit ball in $\mathbb{R}^d$. Let $\mathcal{X}'$ be $\mathcal{X}$ translated so

that its centroid μ is the origin, and scaled so that $\mathcal{X} \subset B_2^d$. There is an ordering $x_1, x_2, \ldots$ of $\mathcal{X}'$ such that $\sum_{i \le t} x_i \in S(B_2^d) \cdot B_2^d$ for all t. This gives a solution to the Permutation Problem on $\mathcal{X}$ with $\alpha = S(B_2^d) \cdot \max_{x \in \mathcal{X}} \|x\|_2$.

It is conjectured that $S(B_2^d) = O(\sqrt{d})$ (see [11, 12]). If true, that would imply a solution to Permutation Problem, and hence to the Sampling Problem, with $\alpha = O(\sqrt{d}) \cdot \max_{x \in \mathcal{X}} \|x\|_2$. This would match our lower bound in Section 5.5.

4.2 Steinitz Algorithm

The best known bound on $S(B_2^d)$ is the Steinitz Lemma: $S(B_2^d) \le d$ ([11]). Furthermore, the proof is algorithmic, so this gives an efficient solution to the Sampling Problem with $\alpha = d \cdot \max_{x \in \mathcal{X}} \|x\|_2$. We call this solution *Steinitz algorithm*. This result is of mild interest, in that it answers the question raised in the previous section on whether there is a solution with error linear in d *without* the various assumptions on $\mathcal{K}$.

We run the Steinitz algorithm on the matrix $\mathcal{F}$ (defined in Section 2.3) to generate 50 pseudosamples. Figure 4.1 shows the $\|\cdot\|_2$ of the empirical error and the theoretical error ($d\delta/t$) for this algorithm. Note that the empirical error of the Steinitz algorithm periodically gets zero. The error starts at zero and after each $|\mathcal{X}|$ iterations, the algorithm adds a complete ordering of $\mathcal{X}$ that contributes zero to the error.

4.3 Our Approach for Solving the Permutation Problem

Due to the connection from the Sampling Problem to the Permutation Problem described in Section 4.1, we will focus on solving Permutation Problem. The main result of this section is Theorem 4.3.1 which solves Permutation Problem with error that is bounded by $O(\sqrt{d}\log^{2.5} n)$.

Theorem 4.3.1. Let $V \subset \mathbb{R}^d$ satisfy $\max_{v \in V} \|v\|_2 \le 1$ and $\sum_{v \in V} v = 0$. Let $n = |V|$. Then there is an ordering $v_1, v_2, \ldots$ of V such that $\|\sum_{i \le t} v_i\|_2 = O(\sqrt{d}\log^{2.5} n)$ for all $t \ge 1$. Furthermore, there is an algorithm with running time $\mathrm{poly}(d, n)$ to find such an ordering.

Although Theorem 4.3.1 implies a solution to Theorem 2.7.2, it does not imply anything about the Steinitz constant $S(B_2^d)$ because that quantity must not depend on n. To explain the proof of Theorem 4.3.1, we must introduce some further definitions and notation [8].

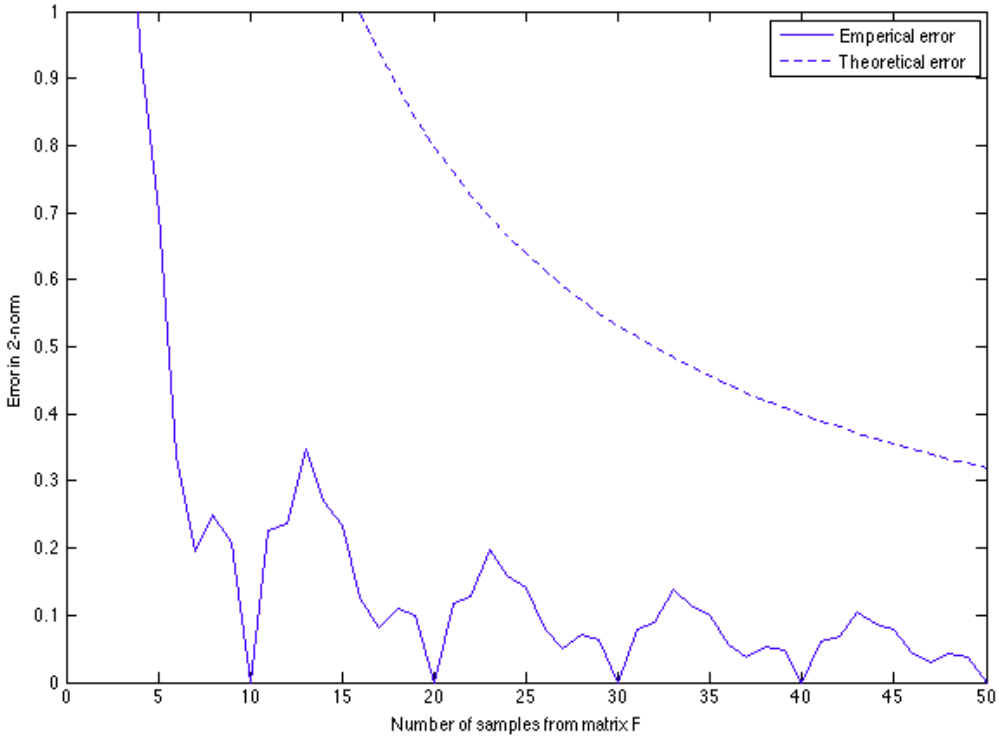

Figure 4.1: Error of the Steinitz algorithm for matrix $\mathcal{F}$ in $\|\cdot\|_2$.

Let $B \subset \mathbb{R}^d$ be a full-dimensional convex body that is 0-symmetric. Let $B' \subset \mathbb{R}^d$ be bounded. For $n \in \mathbb{N}$, we define $\mathbf{sv}(B, B'; n)$, $\mathbf{ss}(B, B'; n)$ and $\mathbf{st}(B, B'; n)$ as follows:

- $\mathbf{sv}(B, B'; n)$ is the smallest $\lambda > 0$ such that for all $x_1, \ldots, x_n \in B'$ there exist $\epsilon_1, \ldots, \epsilon_n \in \{-1, 1\}$ such that $\sum_{i=1}^n \epsilon_i x_i \in \lambda B$. The symbol $\mathbf{sv}$ is abbreviation of "signed vectors".

- $\mathbf{ss}(B, B'; n)$ is the smallest $\lambda > 0$ such that for all $x_1, \ldots, x_n \in B'$ there exist $\epsilon_1, \ldots, \epsilon_n \in \{-1, 1\}$ such that $\sum_{i=1}^k \epsilon_i x_i \in \lambda B$ for all $k \in [n]$. The symbol $\mathbf{ss}$ is abbreviation of "signed series".

- $\mathbf{st}(B, B'; n)$ is the smallest $\lambda > 0$ such that for all $x_1, \ldots, x_n \in B'$ with $\sum_{i=1}^n x_i = 0$ there exists permutation π of $[n]$ such that $\sum_{i=1}^k x_{\pi(i)} \in \lambda B$ for all $k \in [n]$. The symbol $\mathbf{st}$ is abbreviation of "Steinitz constant".

Let us now state some results using this notation. Let B_p be the unit ball of the ℓ_p-norm in $\mathbb{R}^d$. Theorem 4.3.1 states that $\mathbf{st}(B_2, B_2; n) \leq O(\sqrt{d}\log^{2.5} n)$. It is known that $\mathbf{sv}(B_\infty, B_\infty; n) \leq O(\sqrt{d\log(2n/d)})$ and that $\mathbf{sv}(B_\infty, B_1; n) \leq 2$. These results can be found in standard references [1]. It is also known that $\mathbf{sv}(B_\infty, B_2; n) \leq O(\sqrt{\log d})$ [7]. The central open question in this area is Komlós' conjecture that $\mathbf{sv}(B_\infty, B_2; n) \leq O(1)$ [28].

Algorithmic proofs are known for some of the results mentioned above. Recent algorithmic breakthroughs proved that $\mathbf{sv}(B_\infty, B_\infty; n) \leq O(\sqrt{d\log(2n/d)})$ [9, 24]. These techniques also algorithmically prove $\mathbf{sv}(B_\infty, B_2; n) \leq O(\log d)$, which is slightly worse than Banaszczyk's result. The core of these algorithmic results is the following lemma ([24]). We will also use this lemma as a key ingredient in our proof in Section 5.1.

Lemma 4.3.2 (Partial Coloring Lemma). Let $v_1, \ldots, v_d \in \mathbb{R}^n$ be vectors and let $x_0 \in [-1, 1]^n$. Let $c_1, \ldots, c_d \geq 0$ be scalars such that $\sum_{i=1}^d \exp(-{c_i}^2/16) \leq n/16$. Let $\epsilon > 0$. Then there exists a randomized algorithm which with probability at least 0.1 finds a point $x \in [-1, 1]^n$ such that

(i) $|\langle x - x_0, v_i\rangle| \leq c_i\|v_i\|_2$ for all $i \in [d]$

(ii) $|x_j| \geq 1 - \epsilon$ for at least $n/2$ indices $j \in [n]$.

The running time of the algorithm is $\mathrm{poly}(n, d, 1/\epsilon)$.

Now that we have introduced the essential tools from the discrepancy theory, we are ready to explain our main result in this new framework. In the next chapter we prove Theorem 2.7.2; this is equivalent to proving $\mathbf{st}(B_2, B_2; n) \leq O(\sqrt{d}\delta\log^{2.5} n)$ constructively.

Chapter 5

Our Algorithm

In this chapter we present the main result of this thesis which is an efficient randomized algorithm for solving Sampling Problem. Our goal is to give an algorithmic proof for $\mathbf{st}(B_2, B_2; n) \leq O(\sqrt{d}\delta \log^{2.5} n)$. We can briefly explain our approach through the following steps:

- $\diamond 1$ We use an iterative application of the partial coloring lemma (Lemma 4.3.2) to show that $\mathbf{ss}(B_\infty, B_2; n) \leq O(\log^{2.5} n)$. The procedure is explained as the proof of Theorem 5.1.1.
- $\diamond 2$ We show that there is an algorithmic reduction from $\mathbf{st}$ to $\mathbf{ss}$. Formally, we give a constructive proof for $\mathbf{st}(B_\infty, B_2; n) \leq \mathbf{ss}(B_\infty, B_2; n)$ in Theorem 5.2.1. Our proof is an algorithmic version of the proof of a classic theorem in discrepancy theory.
- $\diamond 3$ Applying some standard norm inequalities we conclude that $\mathbf{st}(B_2, B_2; n) \leq O(\sqrt{d} \log^{2.5} n)$. This completes the proof of Theorem 4.3.1.

Once we have algorithmic proof for $\mathbf{st}(B_2, B_2; n) \leq O(\sqrt{d} \log^{2.5} n)$ it is then straightforward to prove Theorem 2.7.2; use this algorithm to get an ordering of elements of $\mathcal{X}$ and repeat this ordering for infinitely many times.

The rest of this chapter is organized as follows. In Section 5.5 and Section 5.6 we prove Theorem 2.7.1 and Theorem 2.7.3 respectively. We finish this part by comparing empirical results of our algorithm, Herding algorithm and Random sampling.

5.1 Bounding $\mathbf{ss}$

In this section, we will construct the first step of our algorithm.

Theorem 5.1.1. Let V be a matrix of size $d \times n$ such that $\max_j \|V_{*,j}\|_2 \leq \delta$. Then there exists $\chi \in \{-1, 1\}^n$ such that $\|\sum_{j=1}^k V_{*,j}\chi_j\|_\infty \leq O(\delta \log^{2.5} n)$ for all

$k \in [n]$. Furthermore, there exists a randomized algorithm to compute the coloring χ in time $\mathrm{poly}(n, d)$.

Proof sketch. The partial coloring lemma can be used iteratively to control the discrepancy of the rows of matrix V; this is the standard technique of Spencer used to bound $\mathbf{sv}(B_\infty, B_2; n)$. However, it does not suffice to bound $\mathbf{ss}(B_\infty, B_2; n)$. We must additionally control the discrepancy of all *prefixes* of rows of V. The idea is to construct from V a new matrix W by adding rows that indirectly control the prefixes of rows of V. Some care is required to ensure that δ does not increase dramatically. Finally, we iteratively apply the partial coloring lemma to W to obtain the desired coloring.

Proof. The proof has several steps:

Construction of W. We construct a new matrix W from V by adding new rows which are "subrows" of the rows of V. Roughly speaking, for every row of V, we make many new copies of that row. Each copy has most of its entries zeroed out, except those in a contiguous region of length 2^k that ends at a multiple of 2^k.

More formally, define $\mathcal{I} := \left\{ [j \cdot 2^k + 1, (j+1) \cdot 2^k] \cap [n] \ : \ j \geq 0, k \geq 0 \right\}$, and note that $|\mathcal{I}| \leq 2n - 1$. We may enumerate the entries of $\mathcal{I}$ as $I_1, I_2, \ldots$. For each $i \in [d]$, let W^i be a matrix of size $|\mathcal{I}| \times n$, constructed from the row $V_{i,*}$ as follows.

$$\forall a \in \{1, \ldots, |\mathcal{I}|\} \qquad W^i_{a,l} \ = \ \begin{cases} V_{i,l} & \text{if } l \in I_a \\ 0 & \text{otherwise} \end{cases}$$

Note that $V_{i,*}$ itself is one of the rows of W^i. Finally, the matrix W is constructed by "stacking" the matrices W^i on top of each other. That is, the rows of W are precisely the set of rows that appear in any W^i. So the matrix W has size $d' \times n$, where $d' = d \cdot |\mathcal{I}| = O(nd)$.

Let δ' be $\max_j \|W_{*,j}\|_2$, the largest 2-norm of any column of W. We claim that $\delta' \leq \delta \cdot \sqrt{\lceil \log n \rceil}$. To see this, note that for every $l \in [n]$, we have $|\{ \ I \in \mathcal{I} \ : \ l \in I \ \}| \leq \lceil \log n \rceil$. Thus, each entry $V_{i,j}$ appears in at most $\lceil \log n \rceil$ of the copies of row $V_{i,*}$.

Applying the partial coloring lemma. The next step of the proof is to apply the partial coloring lemma to the matrix W. We set the parameter $c_i = \delta' / \|W_{i,*}\|_2$, where $C = 32$ and δ' is the maximum 2-norm of any column of W.

We now check the hypotheses of the partial coloring lemma. The argument is similar to existing arguments [9, 24]. We give a complete proof of this in.

We have

$$\sum_{i=1}^{d'} \|W_{i,*}\|_2^2 \;=\; \sum_{j=1}^{n} \|W_{*,j}\|_2^2 \;\le\; n\delta'^2.$$

In particular, there are at most $n/2^r$ rows $W_{i,*}$ with $\|W_{i,*}\|_2^2 \in [2^r\delta'^2, 2^{r+1}\delta'^2]$. So

$$\sum_{i=1}^{d'} \exp\left(\frac{-{c_i}^2}{16}\right) \;<\; \sum_{r\in\mathbb{Z}} \frac{n}{2^r} \exp\left(\frac{-C^2}{2^{r+5}}\right). \tag{5.1.1}$$

We prove in Section 5.8 that the right-hand side 5.1.1 is at most $n/16$ if $C = 32$. This establishes the hypothesis of partial coloring lemma.

Iteratively applying the partial coloring lemma. This step is identical to previous result [9, 24]. The idea is that a partial coloring ensures that at least half of the coordinates are very close to either 1 or -1. If we recurse on the remaining coordinates we can obtain a full coloring. The depth of the recursion is only $\lceil \log n \rceil$ since the number of remaining coordinates halves in each step.

Each application of the partial coloring lemma increases the discrepancy of row $W_{i,*}$ by at most $c_i \|W_{i,*}\|_2 = C\delta'$. The final vector resulting from this recursion is a vector $\chi \in [-1, 1]^n$ satisfying $|\chi_j| \ge 1 - \varepsilon$ for all $j \in [n]$, and:

$$|\langle W_{i,*}, \chi \rangle| \;\le\; C\delta' \cdot \lceil \log n \rceil \;=\; O(\delta' \log n) \qquad \forall i \in [d']. \tag{5.1.2}$$

For further explanation, see [24].

Rounding. The next step is to produce an integral coloring. As in prior work, we simply round each coordinate of χ to either $+1$ or -1. The additional discrepancy introduced by this rounding can be easily bounded. Every entry $V_{i,j}$ satisfies $V_{i,j} \le \|V_{*,j}\|_2 \le \delta$. Increasing $|x_i|$ from $1 - \varepsilon$ to 1 changes the discrepancy by at most $\varepsilon\delta n$. Choosing $\varepsilon \le 1/n\delta$, the additional discrepancy produced by this rounding is at most 1.

Discrepancy of prefixes. Finally, we must show that every prefix of every row of V has low discrepancy under the coloring χ. Observe that any set $P = \{1, \ldots, k\}$, $k \le n$, can be written as the disjoint union $P = \cup_{a\in S} I_a$ for some $S \subseteq [\,|\mathcal{I}|\,]$ with $|S| \le O(\log n)$. Thus, for any $i \in [d]$,

$$\begin{aligned} |\textstyle\sum_{j=1}^{k} V_{i,j}\chi_j| &\le \textstyle\sum_{a\in S} |\sum_{j\in I_a} V_{i,j}\chi_j| \\ &= \textstyle\sum_{a\in S} |\langle\, W^i_{a,*},\, \chi\,\rangle| \\ &\le |S|\cdot O(\delta' \log n) \\ &= O(\delta' \log^2 n) \;=\; O(\delta \log^{2.5} n). \end{aligned}$$

This completes the proof. □

5.2 Relating ss and st

In the previous section we gave an algorithmic proof of $\mathbf{ss}(B_\infty, B_2; n) \le O(\log^{2.5} n)$. The next step in proving Theorem 4.3.1 is to relate ss to st; furthermore, the proof should be *constructive*. The *Chobanyan* theorem is a classic theorem in discrepancy that relates ss and st. Unfortunately, the standard proof of this theorem (see [11]) is not constructive. We derive an algorithmic version of the Chobanyan theorem, leading to the following result.

Theorem 5.2.1. Let V be a real matrix of size $d \times n$ with $\sum_j V_{*,j} = 0$. Let $\delta = \max_j \|V_{*,j}\|_2$. Assume that for every permutation π of $[n]$ there exist $\epsilon_1, \ldots, \epsilon_n \in \{-1, 1\}$ (depending on π) such that $\max_{1\le k\le n} \|\sum_{j=1}^{k} \epsilon_j V_{*,\pi(j)}\|_\infty \le A$. Furthermore, assume that there is an algorithm with running time $\mathrm{poly}(n, d)$ to compute the signs $\epsilon_1, \ldots, \epsilon_n$. Then there is an algorithm to construct a permutation π^* of $[n]$ such that $\max_{1\le k\le n} \|\sum_{j=1}^{k} V_{*,\pi^*(j)}\|_\infty \le A + \delta$. This algorithm also has running time $\mathrm{poly}(n, d)$.

Proof. We describe an algorithm that outputs the desired permutation π^*. For each permutation π of $[n]$, define $\mathrm{disc}(\pi) = \max_{1\le k\le n} \|\sum_{j=1}^{k} V_{*,\pi(j)}\|_\infty$.

Consider an arbitrary permutation π_1 of $[n]$, and let $B_1 = \mathrm{disc}(\pi_1)$. If $B_1 \le A$ then simply output π_1. Otherwise, by the hypothesis of the theorem, there exist $\epsilon^1_1, \ldots, \epsilon^1_n \in \{-1, 1\}$ such that $\max_{1\le k\le n} \|\sum_{j=1}^{k} \epsilon^1_j V_{*,\pi_1(j)}\|_\infty \le A$. We construct permutation π_2 from π_1 as follows. Let

$$M^+ \;=\; \left\{\, j \in [n] \,:\, \epsilon^1_j = +1 \,\right\} \qquad \text{and} \qquad M^- \;=\; \left\{\, j \in [n] \,:\, \epsilon^1_j = -1 \,\right\}.$$

We have

$$\begin{aligned} \textstyle\sum_{j=1}^{k} V_{*,\pi_1(j)} \;+\; \sum_{j=1}^{k} \epsilon^1_j V_{*,\pi_1(j)} \;&=\; 2\cdot \textstyle\sum_{j\in M^+\cap[k]} V_{*,\pi_1(j)} \\ \textstyle\sum_{j=1}^{k} V_{*,\pi_1(j)} \;-\; \sum_{j=1}^{k} \epsilon^1_j V_{*,\pi_1(j)} \;&=\; 2\cdot \textstyle\sum_{j\in M^-\cap[k]} V_{*,\pi_1(j)}. \end{aligned}$$

Hence

$$\begin{aligned}\|\textstyle\sum_{j\in M^+\cap[k]} V_{*,\pi_1(j)}\|_\infty &\le \frac{A+B_1}{2}\\ \|\textstyle\sum_{j\in M^-\cap[k]} V_{*,\pi_1(j)}\|_\infty &\le \frac{A+B_1}{2}.\end{aligned}$$

The new permutation π_2 is formed by concatenating the elements of M^+, in the order given by π_1, followed by the elements of M^-, in the *reverse* of the order given by π_1. It follows from the assumption $\sum_j V_{*,j} = 0$ that

$$\max_{1\le k\le n} \|\textstyle\sum_{j=1}^{k} V_{*,\pi_2(j)}\|_\infty \le \frac{A+B_1}{2}. \tag{5.2.1}$$

Let $B_2 := \mathrm{disc}(\pi_2)$. If $B_2 \le A$, we output π_2. Otherwise, (5.2.1) gives $B_2 \le \frac{A+B_1}{2}$, and so $A < B_2 < \frac{A+B_1}{2} < B_1$. We then repeat the procedure explained above to get another ordering π_3 from π_2.

Let $\ell = \log\big((B_1 - A)/\delta\big)$ and suppose we repeat this process ℓ times. Possibly some ordering π_i has $\mathrm{disc}(\pi_i) \le A$, in which case the algorithm sets $\pi^* = \pi_i$ and terminates. If not, then since $B_{i+1} - A < (B_i - A)/2$, we have $\mathrm{disc}(\pi_\ell) = B_\ell \le A + \delta$. So the algorithm sets $\pi^* = \pi_\ell$ and terminates.

As a final remark, we should specify how to choose the initial ordering π_1. A good choice is to use the ordering produced by the algorithmic proof of the Steinitz lemma [11]. This choice ensures that $B_1 = d\delta$, and so the number of iterations ℓ is at most $\log(B_1/\delta) \le \log d$. □

5.3 Proofs of Main Results

In this section, we will explain the proofs of Theorem 4.3.1 and Theorem 2.7.2. The previous two sections have presented algorithmic proofs that $\mathbf{ss}(B_\infty, B_2; n) \le O(\log^{2.5}(n))$ and that $\mathbf{st}(B_\infty, B_2; n) \le \mathbf{ss}(B_\infty, B_2; n)$. The proof of Theorem 4.3.1 now follows easily.

Proof. (of Theorem 4.3.1) We may think of the set V as a $d \times n$ matrix by imposing an arbitrary order on the vectors. Applying Theorem 5.1.1 with $\delta = 1$ shows that there exists an efficient randomized algorithm that, for any ordering on the columns of V, computes a coloring $\chi \in \{-1, 1\}^n$ with $\max_{1\le k\le n} \|\sum_{j=1}^{k} V_{*,j}\chi_j\|_\infty \le O(\log^{2.5} n)$.

The existence of this algorithm satisfies the key hypothesis of Theorem 5.2.1. Consequently Theorem 5.2.1 asserts that there exists an algorithm with running time $\mathrm{poly}(n, d)$ that constructs a permutation π of these vectors with

$$\max_{1\le t\le n} \|\textstyle\sum_{j=1}^{t} V_{*,\pi(j)}\|_\infty \;\le\; O(\log^{2.5} n). \tag{5.3.1}$$

Using standard norm inequalities, we have that

$$\max_{1\le t\le n} \|\textstyle\sum_{j=1}^{t} V_{*,\pi(j)}\|_2 \;\le\; O(\sqrt{d}\log^{2.5} n),$$

as required. □

As observed earlier, Theorem 4.3.1 easily implies Theorem 2.7.2.

Proof. (of Theorem 2.7.2)

Let $\mathcal{X} \subset \mathbb{R}^d$ and let us denote its members as $x_1, \ldots, x_n$. Let $y_1, y_2, \ldots$ be the infinite sequence computed by Algorithm 2. We must show that this sequence has error $O\big(\sqrt{d}\log^{2.5}(n)/t\big)$.

Consider any $t \ge 0$, and let p and $0 \le r \le n-1$ be integers such that $t = pn + r$. Note that $\sum_{j=1}^{n} x_j - n\mu = 0$. Consequently, the error of $y_0, \ldots, y_t$ is

$$\begin{aligned}
\|\textstyle\sum_{j=1}^{t} y_j/t - \mu\|_2 \;&=\; \|\textstyle\sum_{j=1}^{t} y_j - t\mu\|_2/t \\
&=\; \|(\textstyle\sum_{j=1}^{pn} y_j - pn\mu) \;+\; (\sum_{j=pn+1}^{t} y_j - r\mu)\|_2/t \\
&=\; \|p(\underbrace{\textstyle\sum_{j=1}^{n} x_j - n\mu}_{=0}) \;+\; (\textstyle\sum_{j=1}^{r} x_{\pi(j)} - r\mu)\|_2/t \\
&=\; \|\textstyle\sum_{j=1}^{r} x_{\pi(j)} - r\mu\|_2/t
\end{aligned}$$

By Theorem 4.3.1, the ordering π satisfies

$$\begin{aligned}
&\max_{1\le r\le n} \|\textstyle\sum_{j=1}^{r} v_{\pi(j)}\|_2 \;\le\; O(\sqrt{d}\log^{2.5} n) \\
\implies \qquad &\max_{1\le r\le n} \|\textstyle\sum_{j=1}^{r} x_{\pi(j)} - t\mu\|_2 \;\le\; O(\sqrt{d}\delta\log^{2.5} n).
\end{aligned}$$

It follows that error is

$$\|\textstyle\sum_{j=1}^{t} y_j/t - \mu\|_2 \;\le\; O\big(\sqrt{d}\delta\log^{2.5}(n)/t\big),$$

as required.

□

Algorithm 2 Our new algorithm for the Sampling Problem

Input: a set $\mathcal{X} = \{x_1, \ldots, x_n\} \subseteq \mathbb{R}^d$.

Let $\delta = \max_{i\in[n]} \|x_i\|_2$ and $\mu = \sum_{i\in[n]} x_i/n$.

Let $V = \{v_1, \ldots, v_n\}$ where $v_i = (x_i - \mu)/2\delta$.

Since $\sum_{i\in[n]} v_i = 0$ and $\max_{i\in[n]} \|v_i\|_2 \leq 1$, we may apply the algorithm of Theorem 4.3.1 to get an ordering $\pi : [n] \to [n]$ of the elements of V.

Output: The infinite sequence $y_1, y_2, \ldots$ obtained by ordering the vectors in $\mathcal{X}$ according to the ordering π, and then repeating this ordering infinitely many times. Formally, for all $t \geq 1$, the t^{th} output point is $y_t = x_{\pi(t \mod n)}$, if we assume that the range of $\mod n$ is $\{1, \ldots, n\}$.

5.4 Performance of Our Algorithm

We run our algorithm on matrix $\mathcal{F}$ with $d = 5$ and $n = 10$ to generate 50 pseudosamples. We will measure the error in the ℓ_2-norm and the ℓ_∞-norm. The cyclic behaviour in both graphs shows the fact that the error of algorithm periodically hits zero every $|\mathcal{X}|$ steps. The zero error appears each time that a permutation of $\mathcal{X}$ is added to the sequence of samples generated so far. Let's Compare this figure with Figure 2.2, Figure 2.3 and Figure 4.1 for the ℓ_2-norm error on matrix $\mathcal{F}$, it can be seen that our algorithm has worse performance than Random sampling, the Herding algorithm and the Steinitz algorithm. In Section 5.6 we will show that our algorithm beats the Herding algorithm, if we measure the error in the ℓ_∞-norm.

5.5 Lower Bound for the Sampling Problem

In this section we prove that *every* algorithm for the Sampling Problem has error $\Omega(\sqrt{d}\delta/t)$, proving Theorem 2.7.1. This shows that the algorithm of Theorem 2.7.2 is optimal, up to logarithmic factors.

Theorem 5.5.1. For any d, let $n = d^2$ and define a matrix V of size $d \times (n + d)$ by

$$V_{d\times(n+d)} = \begin{bmatrix} -1/n & \cdots & -1/n & 1 & \cdots & & 0 \\ -1/n & \cdots & -1/n & \vdots & \ddots & & \vdots \\ \vdots & & \vdots & & & 1 & 0 \\ -1/n & \cdots & -1/n & 0 & \cdots & 0 & 1 \end{bmatrix}.$$

Here, the first n columns are $-1/n \cdot \mathbf{1}$ and the last d columns are $e_1, \ldots, e_d$. Then,

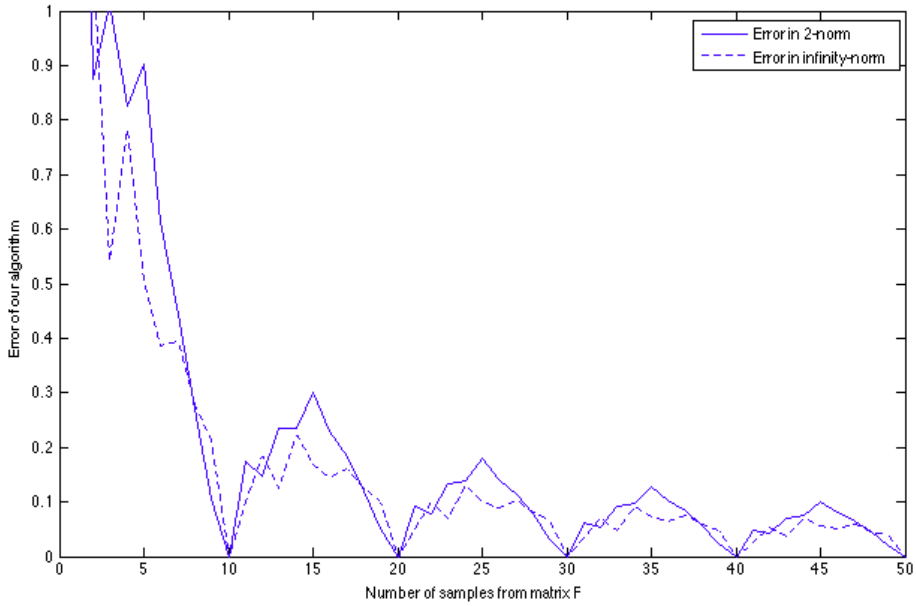

Figure 5.1: Error of our algorithm on matrix $\mathcal{F}$ measured in $\|\cdot\|_2$ and $\|\cdot\|_\infty$.

for any sequence $x_1, x_2, \ldots$ generated from columns of this matrix, there exists $t \in \mathbb{N}$ such that $\|\sum_{i=1}^{t} x_t\|_2 \geq \Omega(\sqrt{d})$.

Proof. Consider any sequence $x_1, x_2, \ldots$. Fix $t = n/2$ and consider the first t elements $x_1, \ldots, x_t$ in the sequence. Let a_i be the number of occurrences of e_i in these t elements, and let $a = \sum_{i=1}^{d} a_i$. Define $z := \sum_{i=1}^{t} x_i$. Then

$$z \;=\; \begin{bmatrix} a_1 - (t-a)/n \\ a_2 - (t-a)/n \\ \vdots \\ a_d - (t-a)/n \end{bmatrix}$$

The i^{th} coordinate of z is equal to $a_i - 1/2 + a/n$. We consider two cases:

Case 1: $a \geq n/3$. Then, there exists an index l for which $a_l \geq n/3d$. So

$$z_l \;\geq\; n/3d - 1/2 + 1/3 \;=\; n/3d - 1/6 \;\geq\; d/3 - 1/6.$$

Thus $\|z\|_2 \geq \Omega(d) = \Omega(\sqrt{d})$.

Case 2: $a \leq n/3$. Then, for all $i \in [d]$

$$a_i - 1/2 \;\leq\; z_i \;\leq\; a_i - 1/2 + 1/3 \;=\; a_i - 1/6.$$

Note that $a_i \in \mathbb{N}$ so $|z_i| \geq 1/6$. So every coordinate of z has absolute value greater than $1/6$. Thus

$$\|\textstyle\sum_{i=1}^{t} x_t\|_2 \;=\; \|z\|_2 \;=\; \Omega(\sqrt{d}),$$

as required. □

Proof. (of Theorem 2.7.1) Let $\mathcal{X}$ consist of the columns of V. Note that $\mu = \sum_{x \in X} x = 0$, and that $\delta = \max_{x \in X} \|x\|_2 = 1$. Assume that we have an algorithm that solves the Sampling Problem on $\mathcal{X}$ and generates pseudosamples $x_1, x_2, \ldots$. By Theorem 5.5.1 there exists an index $t \in \mathbb{N}$ such that $\|\sum_{i=1}^{t} x_t\|_2 \geq \Omega(\sqrt{d})$. Therefore

$$\|\textstyle\sum_{i=1}^{t} x_i/t - \mu\|_2 \;\geq\; \Omega(\sqrt{d}\delta/t),$$

as required. □

5.6 $\|\cdot\|_\infty$ Lower Bound for the Herding Algorithm

In this section, we define matrix $\mathcal{J}$ and prove that the error of Herding algorithm on this data set gets bigger than $\Omega(\sqrt{d})$, at some point.

Theorem 5.6.1. Let matrix $\mathcal{J}_{d \times n}$ be defined as

$$\left[\begin{array}{c|ccc|ccc|cccc|ccc}
\overbrace{\phantom{1/\sqrt{d}}}^{A} & \overbrace{}^{B} & & & \overbrace{}^{C} & & & \overbrace{}^{D} & & & & \overbrace{}^{E} & & \\
1/\sqrt{d} & -\epsilon & \cdots & -\epsilon & 0 & \cdots & 0 & -1 & 0 & \cdots & 0 & \epsilon & \cdots & \epsilon \\
 & & & & & & & 0 & \ddots & & & & & \\
\vdots & \vdots & & \vdots & & & & \vdots & & & 0 & \vdots & & \\
1/\sqrt{d} & -\epsilon & \cdots & -\epsilon & 0 & \cdots & 0 & 0 & \cdots & 0 & -1 & \epsilon & \cdots & \epsilon \\
1 & 0 & \cdots & 0 & -\epsilon & \cdots & -\epsilon & 0 & \cdots & 0 & 0 & 0 & \cdots & 0
\end{array}\right]$$

in which $|A| = 1$, $|B| = \left\lceil 1/(\sqrt{d}\epsilon) \right\rceil$, $|C| = \lceil 1/\epsilon \rceil$, $|D| = d - 1$, and $|E| = \lceil 1/\epsilon \rceil$. So $n = d + \left\lceil 1/\sqrt{d}\epsilon \right\rceil + \lceil 1/\epsilon \rceil + \lceil 1/\epsilon \rceil$. This definition ensures that μ, the average of the columns, is 0. We define $\epsilon = 1/d\sqrt{d}$. Consider matrix $\mathcal{J}$ and let $x_1, x_2, \ldots$ be the pseudosamples generated by the Herding algorithm when we apply it to matrix V, with $w_0 = 0$. Then there exists $T \in \mathbb{N}$ such that $\|\sum_{i=1}^{T} x_i\|_\infty \geq \Omega(\sqrt{d})$.

Proof. Set $T = (2\sqrt{d}+1)(\sqrt{d}/2-1)$. Assume that the Herding algorithm has chosen pseudosamples $x_1, x_2, \ldots, x_T$ in the first T iterations. From the definition of the Herding algorithm we know that $w_i = w_{i-1} - x_i$ for all $1 \leq i \leq T$. Summing these T equalities $w_T = w_0 - (x_1 + x_2 + \ldots + x_T)$. Substituting $w_0 = 0$, we get that $w_T = -(x_1 + x_2 + \ldots + x_T)$. So in order to compute the value of $\sum_{i=1}^{T} x_i$ we only need to compute the vector w_T.

We claim that for all $1 \leq t \leq T$, w_t is as follows. Choose $0 \leq k \leq \sqrt{d}/2 - 1$ such that $(2\sqrt{d}+1)(k-1)+1 < t \leq (2\sqrt{d}+1)k+1$ then

$$w_t = \big[\overbrace{1-k/\sqrt{d}, \cdots, 1-k/\sqrt{d}}^{t-1}, \overbrace{-k/\sqrt{d}, \cdots, -k/\sqrt{d}}^{d-t}, \overbrace{-k}^{1}\big]^\mathsf{T}$$

in which there are $t-1$ coordinates of value $1-k/\sqrt{d}$ on the top, $d-t$ coordinates of value $-k/\sqrt{d}$ in the middle and one coordinate of value $-k$ at the bottom. We prove our claim by induction:

Initial case of induction; $t = 1$. From the definition of the Herding algorithm, x_1 is defined to be $\arg\max_{j\in[n]} \langle w_0, V_{*,j}\rangle$. Since $w_0 = 0, x_1 = V_1$. So $w_1 = w_0 - x_1 = -V_1$. This is the same as w_1 in the above formula for $k = 1$.

Inductive step. Assuming that the hypothesis is true for some $t = 1, 2, \ldots, l$, we prove that it is also true for $t = l+1$. We consider two cases:

- $t = (2\sqrt{d}+1)k+1$ for some $0 \leq k \leq \sqrt{d}/2 - 1$. In order to get x_{t+1} we need to compute $\langle w_t, V_j\rangle$ for all $j \in [n]$ and get the column that maximizes this value. We consider five cases:

  Case 1: $V_j \in A$. In this case $\langle w_t, V_j\rangle = k/\sqrt{d} + k/d$.
  Case 2: $V_j \in B$. In this case $\langle w_t, V_j\rangle = -\epsilon k(\sqrt{d} + 1 + 1/\sqrt{d})$.
  Case 3: $V_j \in C$. In this case $\langle w_t, V_j\rangle = k\epsilon$.
  Case 4: $V_j \in D$. In this case $\langle w_t, V_j\rangle = -1 + k/\sqrt{d}$ or $k/\sqrt{d}$.
  Case 5: $V_j \in E$. In this case $\langle w_t, V_j\rangle = \epsilon k(\sqrt{d} + 1 + 1/\sqrt{d})$.

  Substituting the values of ϵ, ϵ and ϵ, it is easy to check that the maximum value is achieved by choosing $x_{t+1} = V_1$. So $w_{t+1} = w_t - x_{t+1} =$

  $$\big[\overbrace{1-(k+1)/\sqrt{d}, \cdots, 1-(k+1)/\sqrt{d}}^{t-1}, \overbrace{-(k+1)/\sqrt{d}, \cdots, -(k+1)/\sqrt{d}}^{d-t}, \overbrace{-(k+1)}^{1}\big]^\mathsf{T}$$

  Note that $t = (2\sqrt{d}+1)k+1$ which gives $t+1 = (2\sqrt{d}+1)k+2$. Thus $(2\sqrt{d}+1)((k+1)-1)+1 < t < (2\sqrt{d}+1)(k+1)+1$ which gives the same value for w_{t+1} in the hypothesis.

- $(2\sqrt{d}+1)(k-1)+2 \leq t \leq (2\sqrt{d}+1)k$ for some $1 \leq k \leq \sqrt{d}/2-1$. Similar to the previous case, we need to compute $\langle w_t, V_j \rangle$ for all $j \in [n]$ and get the column that maximizes this value. We consider five cases:

  Case 1: $V_j \in A$. In this case $\langle w_t, V_j \rangle \leq \epsilon(k-1)/\sqrt{d} + k/d$.
  Case 2: $V_j \in B$. In this case $\langle w_t, V_j \rangle \leq \sqrt{d}(2-k) - k - k/\sqrt{d}$.
  Case 3: $V_j \in C$. In this case $\langle w_t, V_j \rangle = k\epsilon$.
  Case 4: $V_j \in D$. In this case $\langle w_t, V_j \rangle = -1 + k/\sqrt{d}$ or $k/\sqrt{d}$.
  Case 5: $V_j \in E$. In this case $\langle w_t, V_j \rangle \leq \epsilon(\sqrt{d}k + k + k/\sqrt{d} - 1)$.

  Similar to the previous case, it is easy to check that the maximum value is achieved in the third case by choosing the $x_{t+1} = -e_t$ from block D. (Recall that e_t denotes the t^{th} standard basis vector). So $w_{t+1} = w_t - x_{t+1} =$

$$\big[\overbrace{1-k/\sqrt{d}, \cdots, 1-k/\sqrt{d}}^{t}, \overbrace{1-k/\sqrt{d}, \cdots, -(k+1)/\sqrt{d}}^{d-1-t}, \overbrace{-(k+1)}^{1}\big]^\intercal$$

  in which the top t coordinate are $1-k/\sqrt{d}$ and the middle $d-1-t$ coordinates are $-k/d$ and the last coordinate is $-k$. Note that $t+1$ is still in the same interval as before and therefore the above vector gives the same value for w_{t+1} as in the hypothesis. This completes the proof of our claim.

The proof completes by setting $T = (2\sqrt{d}+1)(\sqrt{d}/2-1)$ and $k = \sqrt{d}/2-1$. In this case $\|w_T\|_\infty \geq \sqrt{d}/2 - 1 = \Omega(\sqrt{d})$. □

We finish this section by showing that Theorem 5.6.1 holds in application. Figure 5.2 shows the performance of Herding algorithm on matrix $\mathcal{J}$ for $d = 19 \times 19$ to generate 600 pseudosamples. The Y-axis shows $\|t \cdot \text{error}\|_\infty$ of the algorithm and the $X-$axis shows the number of samples. The figure clearly shows that the graph eventually hits the line $\sqrt{d}/2$. The reason that we plot $\|t \cdot \text{error}\|_\infty$ instead of $\|\text{error}\|_\infty$ is that we want to see the growth of the error function better.

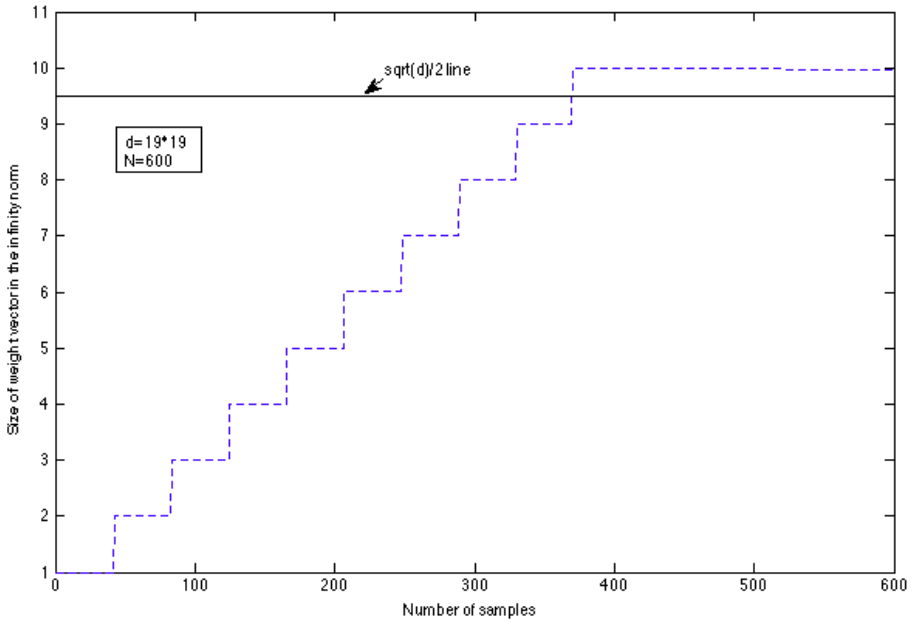

Figure 5.2: Herding algorithm error in $\|\cdot\|_\infty$.

5.7 Comparison

We run our algorithm, Herding algorithm and random sampling on matrices $\mathcal{F}, \mathcal{G}$ with $d = 6$ and $n = 10$ and matrix $\mathcal{J}$ with $d = 6$ to generate 30 pseudosamples. Figure 5.3 shows error of these algorithms measured in $\|\cdot\|_2$ and $\|\cdot\|_\infty$. It can be observed that, our algorithm is doing better than other algorithms if we measure the error in the $\|\cdot\|_\infty$. For the case of $\|\cdot\|_2$, our algorithm is strictly better than random sampling but approximately the same as the Herding algorithm.

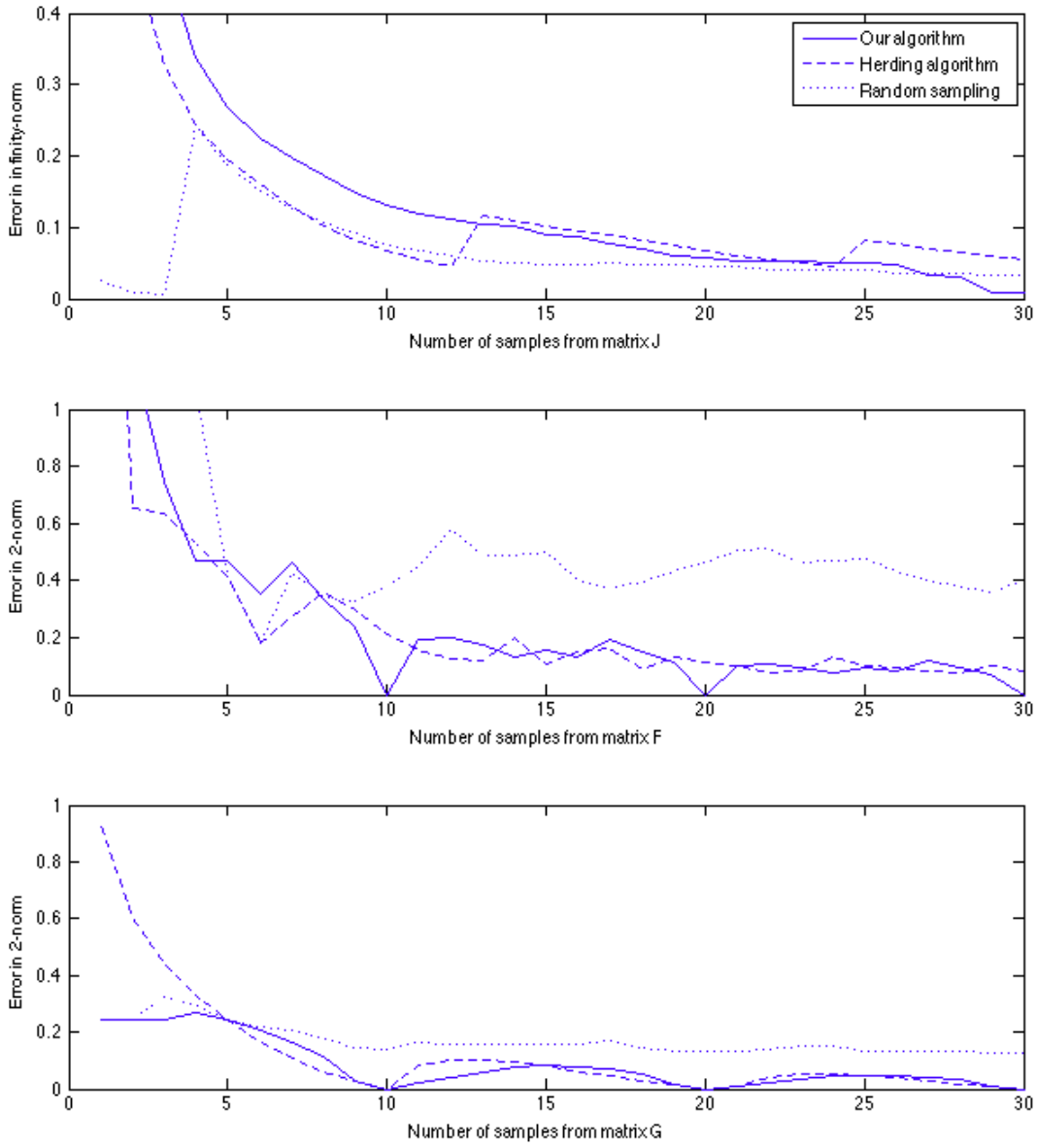

Figure 5.3: Comparison of the error for different sampling algorithms.

5.8 Partial Coloring Lemma Hypothesis

We prove that the right-hand side (5.1.1) is at most $n/16$ if $C = 32$. Dividing by n, it suffices to show that

$$\sum_{r\in\mathbb{Z}} \frac{1}{2^r} \exp\left(\frac{-C^2}{2^{r+5}}\right) \;\le\; 1/16. \tag{5.8.1}$$

First, assuming only that $C \ge 0$, we have

$$\sum_{r\ge 6} \frac{1}{2^r} \exp\left(\frac{-C^2}{2^{r+5}}\right) \;\le\; \sum_{r\ge 6} \frac{1}{2^r} \;=\; 1/32. \tag{5.8.2}$$

Next assume that $C = 32$. Then

$$\begin{aligned}
\sum_{r\le 0} \frac{1}{2^r} \exp\left(\frac{-C^2}{2^{r+5}}\right) \;&=\; \sum_{i\ge 0} 2^i \cdot \exp\left(\frac{-C^2\cdot 2^i}{32}\right) \\
&\le\; \sum_{i\ge 0} \exp\left(i - 32\cdot 2^i\right) \\
&<\; \sum_{i\ge 1} \exp(-31i) \\
&=\; \frac{e^{-31}}{1-e^{-31}} \;<\; 1/200.
\end{aligned} \tag{5.8.3}$$

Finally,

$$\begin{aligned}
\sum_{r=1}^{5} \frac{1}{2^r} \exp\left(\frac{-C^2}{2^{r+5}}\right) \;&=\; \frac{e^{-16}}{2} + \frac{e^{-8}}{4} + \frac{e^{-4}}{8} + \frac{e^{-2}}{16} + \frac{e^{-1}}{32} \\
&<\; 1/44.
\end{aligned} \tag{5.8.4}$$

Summing (5.8.2), (5.8.3) and (5.8.4) shows (5.8.1).

This establishes the hypotheses of the partial coloring lemma.

Chapter 6

Conclusions

In this thesis we studied the theoretical foundation of Herding algorithm and more generally the Sampling Problem. We analyzed lower bound and upper bound analysis for the error of Sampling Problem and designed an efficient algorithm that solves this problem with an error that is near the optimal. To drive our results we used randomness and made several assumptions on the input data. In this chapter, we discuss how these assumptions can be generalized to conduct further research on this problem.

6.1 Remarks

Our algorithm in Chapter 5 is described as a randomized algorithm. It can be derandomized by replacing our use of the the Lovett-Meka algorithm with a derandomized form of Bansal's algorithm [10]. It is also plausible to improve the running time of our algorithm by using the recent results that should be faster than the Lovett-Meka algorithm [29].

Our bound $\mathbf{ss}(B_\infty, B_2; n) = O(\log^{2.5} n)$ can be improved if we do not require an efficient algorithm. It is also known [7] that $\mathbf{sv}(B_\infty, B_2; n) = O(\sqrt{\log n})$. Our results in Chapter 5 show that

$$\mathbf{ss}(B_\infty, B_2; n) \;=\; O(\log^{1.5} n) \cdot \mathbf{sv}(B_\infty, B_2; n). \tag{6.1.1}$$

Combining our result with Banaszczyk's yields a solution to the Sampling Problem with error $O(\delta\sqrt{d}\log^2(n)/t)$, but no known efficient algorithm achieves that bound.

Our lower bound in Theorem 2.7.1 is for the particular case that $t = d^2/2$. It would be interesting to get a similar bound for the cases $t \gg d^2$ and $t \ll d^2$.

6.2 Open Questions

Our work leaves several questions unanswered. We do not prove that our algorithm of Chapter 5 has better error than the Herding algorithm. In particular, it is conceivable that the Herding algorithm has optimal error $O(\delta\sqrt{d}/t)$, matching our lower bound. Alternatively, it is also conceivable that the $O(\delta d/t)$ bound on the Herding algorithm's error is actually tight.

We do not know whether our bound in (6.1.1) is tight. It is conceivable that

$$\mathbf{ss}(B_\infty, B_2; n) \;=\; O(1) \cdot \mathbf{sv}(B_\infty, B_2; n). \tag{6.2.1}$$

It seems that techniques substantially different than ours would be required‘ to prove this.

A major open question in discrepancy theory [11, 12] is whether

$$\mathbf{ss}(B_2, B_2; n) \;\leq\; O(\sqrt{d}). \tag{6.2.2}$$

It would be very interesting if (6.2.1) is true, as then then the Komlós conjecture would imply (6.2.2).

Our algorithm in Chapter 5 assumes that the input set $\mathcal{X}$ is finite. It is interesting to study if we can generalize our algorithm for case of infinite $\mathcal{X}$. It would also be interesting to study if our algorithm gives good results for the intended applications of Herding in Markov random fields.

Our definition of the Sampling Problem states that μ is the mean of $\mathcal{X}$, whereas a more general formulation could incorporate a distribution p on $\mathcal{X}$. It is interesting to see if one can modify our algorithm for the more general case. Here we briefly suggest one possible approach. It seems that the more general case should be reducible to the uniform distribution case. Assume that p assigns probability $p(x_i)$ to each point $x_i \in \mathcal{X}$. If each $p(x_i) = \frac{a_i}{b_i}$ for natural numbers a_i and b_i, then one can set $c = LCM(b_1, \ldots, b_n)$ and generate a new multi-set $\mathcal{Y}$ with $c \cdot \frac{a_i}{b_i}$ copies of x_i. The uniform distribution on $\mathcal{Y}$ is the same as the distribution p on the original set $\mathcal{X}$. Now we only need to feed set $\mathcal{Y}$ to our algorithm. There are two caveats with this approach. First, we have ignored irrational numbers and second, we may have $|\mathcal{Y}| \gg |\mathcal{X}|$. It seems possible to mitigate the runtime increase and to deal with irrational numbers through the use of standard sampling lemmas [2], which can approximate p by a distribution for which c is small.

Bibliography

[1] N. Alon and J. Spencer. *The Probabilistic Method*. John Wiley,, 2000.

[2] I. Althöfer. On sparse approximations to randomized strategies and convex combinations. *Linear Algebra and its Applications*, 199:339–355, 1994.

[3] E. Amaldi and R. Hauser. Boundedness theorems for the relaxation method. *Mathematics of Operations Research*, 30:939–955, 2005.

[4] F. Bach, S. Lacoste-Julien, and G. Obozinsk. On the equivalence between Herding and conditional gradient algorithms. In *Proc. ICML*, 2012.

[5] M. F. Balcan. Lecture notes of 8803 machine learning theory, 2011. manuscript.

[6] K. Ball. An elementary introduction to modern convex geometry. *Random Structures and Algorithms*, 31, 1997.

[7] W. Banaszczyk. Balancing vectors and Gaussian measures of n-dimensional convex bodies. *Random Structures and Algorithms*, 12(4):351–360, 1998.

[8] W. Banaszczyk. On series of signed vectors and their rearrangements. *Random Structures and Algorithms*, 40:301–316, 2012.

[9] N. Bansal. Constructive algorithms for discrepancy minimization. In *FOCS*, 2010.

[10] N. Bansal and J. Spencer. Deterministic discrepancy minimization. *Algorithmica*, 67(4):451–471, 2013.

[11] I. Bańány. On the power of linear dependencies. *Building Bridges, Bolyai Society Mathematical Studies*, 19:31–45, 2008.

[12] V. Bergström. Zwei sätze über ebene vektorpolygone. *Abh. Math. Sem. Univ. Hamburg*, 8:148–152, 1931.

[13] H. D. Block and S. A. Levin. On the boundedness of an iterative procedure for solving a system of linear inequalities. *Proceedings of the American Mathematical Society*, 26:229–235, 1970.

[14] L. Bornn, Y. Chen, N. de Freitas, M. Eskelin, J. Fang, and M. Welling. Herded Gibbs sampling. In *International Conference on Learning Representations*, 2013.

[15] B. Chazelle. *The Discrepancy Method: Randomness and Complexity*. Cambridge University Press, 2000.

[16] S. Chen, J. Dick, and A. B. Owen. Consistency of Markov chain quasi-Monte Carlo on continuous state spaces. *Annals of Statistics*, 39:673–701, 2011.

[17] Y. Chen, M. Welling, and A. Smola. Super-samples from kernel Herding. In *Proc. UAI*, 2010.

[18] F. Rosenblatt. The Perceptron: A probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6), 1958.

[19] A. Gelfand, Y. Chen, L. van der Maaten, and M. Welling. On Herding and the Perceptron Cycling theorem. In *Proc. NIPS*, 2010.

[20] N. Harvey. Lecture notes of CPSC 536N: Randomized algorithms, 2011. manuscript.

[21] N. Harvey and S. Samadi. Near-optimal Herding. In *COLT*, pages 1165–1182, 2014.

[22] A. E. Holroyd and J. Propp. Rotor walks and Markov chains. *Algorithmic Probability and Combinatorics*, 520:105–126, 2010.

[23] F. Huszár and D. Duvenaud. Optimally-weighted Herding is Bayesian quadrature. In *In Proc. UAI*, 2012.

[24] S. Lovett and R. Meka. Constructive discrepancy minimization by walking on the edges. In *FOCS*, 2012.

[25] J. Matousek. *Geometric Discrepancy: An Illustrated Guide*. Springer, 1999.

[26] M.L. Minsky and S. Paper. *Perceptrons; An introduction to computational geometry*. MIT Press, 1969.

[27] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, New York, NY, USA, 1995.

[28] J. Spencer. Six standard deviations suffice. *Trans. Amer. Math. Soc.*, 289:679–706, 1985.

[29] T. Rothvoss. Constructive discrepancy minimization for convex sets. *CoRR*, abs/1404.0339, 2014.

[30] R. Vershynin. Lecture notes in geometric functional analysis, 2009. manuscript.

[31] A. Vilenkin, E. P. S. Shellard, X. Lin, C. Genest, D. L. Banks, G. Molenberghs, D. W. Scott, and J. L. Wang. *Past, Present, and Future of Statistical Science*. CRC Press, 2014.

[32] A. Vilenkin, E. P. S. Shellard, Xihong Lin, C. Genest, D. L. Banks, G. Molenberghs, D. W. Scott, and J. L. Wang. Information theory and statistical mechanics. *Physical Review*, 106:620–630, 1957.

[33] M. Welling. Herding dynamical weights to learn. In *Proceedings of the 26th International Conference on Machine Learning (ICML)*, 2009.