

UC Berkeley

UC Berkeley Electronic Theses and Dissertations

Title

Relaxed Wasserstein, Generative Adversarial Networks, Variational Autoencoders and their applications

Permalink

<https://escholarship.org/uc/item/5v77z2d3>

Author

Yang, Nan

Publication Date

2019

Peer reviewed|Thesis/dissertation

Relaxed Wasserstein, Generative Adversarial Networks, Variational Autoencoders and their applications

by

Nan Yang

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Engineering-Industrial Engineering and Operations Research

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Xin Guo, Chair

Professor Paul Grigas

Professor Anil Aswani

Professor Will Fithian

Summer 2019

Relaxed Wasserstein, Generative Adversarial Networks, Variational Autoencoders and their applications

Copyright 2019
by
Nan Yang

Abstract

Relaxed Wasserstein, Generative Adversarial Networks, Variational Autoencoders and their applications

by

Nan Yang

Doctor of Philosophy in Engineering-Industrial Engineering and Operations Research

University of California, Berkeley

Professor Xin Guo, Chair

Statistical divergences play an important role in many data-driven applications. Two notable examples are Distributionally Robust Optimization (DRO) problems and Generative Adversarial Networks (GANs).

In the first section of my dissertation, we propose a novel class of statistical divergence called Relaxed Wasserstein (RW) divergence, which combines Wasserstein distance and Bregman divergence. We begin with its strong probabilistic properties, and then to illustrate its uses, we introduce Relaxed Wasserstein GANs (RWGANs) and compare it empirically with several state-of-the-art GANs in image generation. We show that it strikes a balance between training speed and image quality. We also discuss the potential use of Relaxed Wasserstein to construct ambiguity sets in DRO problems.

In the second section of my dissertation, we show the application of another type of generative neural network, the Variational AutoEncoder (VAE), to metagenomic binning problems in bioinformatics. Shotgun sequencing is used to produce short reads from DNA sequences in a sample from a microbial community, which could contain thousands species of discovered or unknown microbes. The short reads are then assembled by connecting overlapping subsequences and thus forming longer sequences called contigs. Metagenomic binning is the process of grouping contigs from multiple organisms based on their genomes of origin. We propose a new network structure called MetaAE, which combines compositional and reference-based information in a nonlinear way. We show that this binning algorithm improves the performance of state-of-the-art binners by 20% on two independent synthetic datasets.

To my mom and dad.

Contents

Contents	ii
List of Figures	iii
List of Tables	v
1 Outline	1
2 Background	2
2.1 Bregman Divergence	2
2.2 Wasserstein Distance	6
2.3 Neural Networks	8
2.4 Generative Adversarial Networks	10
2.5 Variational Autoencoders	16
3 Relaxed Wasserstein GANs	20
3.1 Introduction	20
3.2 Relaxed Wasserstein Divergence	21
3.3 Experiments	30
3.4 Discussion	33
4 Properties of Bregman divergence and Choices of RW divergence	45
4.1 More Properties of Bregman Divergence	45
4.2 Choices of ϕ in Relaxed Wasserstein divergence	50
5 Applications	52
5.1 Distributionally Robust Optimization	52
5.2 MetaAutoEncoder in Metagenomic Binning	65
5.3 Simulation of Credit Default Swap Index Transaction Data	85
Bibliography	96

List of Figures

2.1	The bias-variance tradeoff of the k -NN algorithm based on the loss functions $D_\phi(x, y) = e^x - e^y - e^y(x - y)$ and $D_\phi(x, y) = x \log\left(\frac{x}{y}\right) + (1 - x) \log\left(\frac{1-x}{1-y}\right)$	5
2.2	Earthmover interpretation of Wasserstein distance	8
2.3	Structure of an LSTM unit	9
2.4	Network structure of the vanilla GAN	12
2.5	Network structure of the conditional GAN	15
2.6	Structure of a variational autoencoder	19
3.1	The decomposition of W_{D_ϕ} where the solid arrow denotes transformation and the dashed arrows denote the divergences between probability distributions.	25
3.2	Inception scores at the beginning and final stages of training. DCGAN refers to the standard DCGAN generator and MLP refers to an ReLU-MLP with 4 hidden layers and 512 units at each layer.	33
3.3	Sample qualities at different stages of training on MNIST.	36
3.4	Sample qualities at different stages of training on Fashion-MNIST.	38
3.5	Training curves of the negative critic loss at different stages of training on MNIST and Fashion-MNIST. G_{loss} and D_{loss} refer to the loss in generative and discriminative nets, which is plotted in orange and blue lines, respectively.	39
3.6	Sample qualities at the initial stage of training on CIFAR-10.	40
3.7	Sample qualities at the final stage of training on CIFAR-10.	41
3.8	Sample qualities at the initial stage of training on ImageNet.	42
3.9	Sample qualities at the final stage of training on ImageNet.	43
3.10	Training curves at different stages of training. DCGAN refers to the standard DCGAN generator and MLP refers to an ReLU-MLP with 4 hidden layers and 512 units at each layer. G_{loss} and D_{loss} refer to the loss in generative and discriminative nets. The loss in RWGANs is shown to converge consistently while the loss in WGANs-GP tends to diverge as the training progresses. WGANs achieves the lowest variance among the three methods.	44
5.1	Shape of the upper and lower bounds of the objective value using an L^2 ambiguity set.	59
5.2	Metagenomics	67

5.3	Model Structure	68
5.4	Clustering results of MetaAE vs. VAMB on the ActinoMock dataset	73
5.5	Clustering results of MetaAE vs. VAMB on the MetaHIT dataset	73
5.6	Histogram of numbers of pairwise nonzero elements of inter-genome and intra-genome contigs in the NT matrix on the MetaHIT dataset	75
5.7	Histogram of similarities of inter-genome and intra-genome contigs in the NT matrix on the MetaHIT dataset	76
5.8	Histogram of similarities of inter-genome and intra-genome contigs in the TNF matrix on the MetaHIT dataset	77
5.9	Clustering results of MetaAE (with SVD) vs. VAMB on the MetaHIT dataset	78
5.10	Clustering results of MetaAE (with quadratic transform) vs. VAMB on the MetaHIT dataset	79
5.11	Clustering results of TNF vs. encoded TNF on the MetaHIT dataset	79
5.12	Clustering results of encoded NT vs. encoded NT and TNF on the MetaHIT dataset	80
5.13	Clustering results of encoded NT vs. MetaAE on the ActinoMock dataset	80
5.14	Histogram of similarities of inter-genome and intra-genome contigs in the AA matrix on the MetaHIT dataset	81
5.15	Histogram of similarities of inter-genome and intra-genome contigs in the NT100 matrix on the MetaHIT dataset	82
5.16	Histogram of similarities of inter-genome and intra-genome contigs in the NT99 matrix on the MetaHIT dataset	82
5.17	Clustering results using only encoded NT99 and encoded NT100 matrix on the MetaHIT dataset	83
5.18	Histogram of similarities of inter-genome and intra-genome contigs in the AA100 matrix on the MetaHIT dataset	83
5.19	Histogram of similarities of inter-genome and intra-genome contigs in the AA99 matrix on the MetaHIT dataset	84
5.20	Structure of SeqGAN	90
5.21	Pie chart of the action variable on the generated and the original datasets	91
5.22	Pie chart of the collateral variable on the generated and the original datasets	91
5.23	Pie chart of the price forming variable on the generated and the original datasets	92
5.24	Histograms of the rounded notional amount variable on the generated and the original datasets	92
5.25	Heatmaps of the pairwise correlation coefficients on the generated and the original datasets	93

List of Tables

5.1	Input Matrices	69
5.2	Sparsity in the MetaHIT dataset	74
5.3	Reconstruction Errors of the MetaHIT dataset using Autoencoders	74
5.4	Confusion matrix on MetaHIT using NT	76
5.5	Confusion matrix on MetaHIT using encoded NT	77
5.6	Accuracy of models predicting the cleared variable on the original and the synthetic datasets	94
5.7	Accuracy of models predicting the price_formulation variable on the original and the synthetic datasets	94
5.8	Accuracy of models predicting the price variable on the original and the synthetic datasets	94

Acknowledgments

I would like to first thank my adviser, Professor Xin Guo. She has guided me with patience, wisdom and kindness throughout the years. I have learned a lot from her, both in my research and in my life. I would also like to thank Professor Paul Grigas, Professor Anil Aswani and Professor Will Fithian for being in my qualifying exam and dissertation committee. Their professional feedback are vital for the completion of my degree.

I would show my gratitude to my collaborators: Tianyi Lin and Johnny Hong from Berkeley, and Zhong Wang and Volkan Sevim at the Joint Genome Institute. Being able to work with them is a precious experience in my life.

I would like to thank my classmates in IEOR: Sheng, Quico, Haoyang, Erik, Shiman, Renyuan, Xu, Ying, and my friends outside of IEOR: Haoyu Wu, Yishuang Chen, Jessica Shui, Jieqi Sun, Qifan Pu, Zenan Wang, Weijia Li, Mujun Zhou, Hong Shang, Zhenxing Zhang and Chenzhe Tian. The time spent with them at Berkeley makes these years enjoyable and memorable.

Last but not least, I would like to thank my parents, Yumin Yang and Changyao Zhang, for their continuous support and unconditional love.

Chapter 1

Outline

This dissertation is organized as follows: we first review the concept and properties of the family of Bregman divergence and the basics of two generative models: variational autoencoders (VAE) and generative adversarial neural networks (GANs) in Chapter 2. In Chapter 3, we propose Relaxed Wasserstein divergences, a new family of divergence functions that enjoys the benefits of both Bregman divergences and Wasserstein distance, and talk about their usefulness in GANs. In Chapter 4, we show how to choose the appropriate underlying convex function for Bregman divergence and Relaxed Wasserstein divergence for different applications. Finally in Chapter 5, we cover three examples of the applications of Bregman divergence and Relaxed Wasserstein divergence to robust optimization problems, variational autoencoders to metagenomic binning problems in bioinformatics, and GANs to data augmentation in finance.

Chapter 2

Background

In this chapter, we first introduce the definition and basic properties of two statistical divergences: Bregman divergence and Wasserstein distance. Then we review the basic ideas behind Generative Adversarial Networks, their applications, and mathematical details. Finally, we illustrate the concept of Variational Autoencoders.

2.1 Bregman Divergence

Bregman divergences (Bregman 1967) were introduced by Lev Bregman in 1967 in solving a problem in convex optimization and then continued by Censor and Lent (1981) and De Pierro and Iusem (1986). Since their inception, Bregman divergences have found applications not only in convex optimization, but also in statistics and machine learning. For example, clustering (Lucic, Bachem, and A. Krause 2015; Banerjee, Merugu, et al. 2005), inverse problems (Le Besnerais, Bercher, and Demoment 1999; Jones and Byrne 1990), classification (Srivastava, Gupta, and Frigiyik 2007), logistic regression and AdaBoost (Collins, Schapire, and Singer 2002; Murata et al. 2004; Lafferty 1999), regression (Kivinen and Warmuth 1998), mirror descent (A. S. Nemirovski 1983), and generalized accelerated descent algorithms (Wibisono and Wilson 2016; Taskar, Lacoste-Julien, and Jordan 2006).

Definition 2.1.1. For two vectors x and y in \mathbb{R}^d and a strictly convex function $\phi(x) : \mathbb{R}^d \rightarrow \mathbb{R}$, the Bregman divergence is defined as

$$D_\phi(x, y) = \phi(x) - \phi(y) - \langle \nabla \phi(y), x - y \rangle.$$

For two continuous distributions \mathbb{P} and \mathbb{Q} , one can define Bregman divergence as in Jones and Byrne (1990),

$$\begin{aligned} D_\phi(\mathbb{P}, \mathbb{Q}) &= \int [\phi(p(x)) - \phi(q(x)) - \phi'(q(x))(p(x) - q(x))] d\mu(x), \end{aligned}$$

where $p(x)$ and $q(x)$ are probability density functions of \mathbb{P} and \mathbb{Q} respectively, μ is the base measure, and $\phi : \mathbb{R} \rightarrow \mathbb{R}$ is a strictly convex function.

For two discrete distributions \mathbb{P} and \mathbb{Q} with common support, one can similarly define Bregman divergence to be

$$D_\phi(\mathbb{P}, \mathbb{Q}) = \sum_{i=1}^d [\phi(p_i) - \phi(q_i) - \nabla\phi(q_i)(p_i - q_i)],$$

where $\{p_i\}_{i=1}^d$ and $\{q_i\}_{i=1}^d$ represent the distributions of \mathbb{P} and \mathbb{Q} respectively.

Examples of Bregman divergences include:

- L^2 divergence: $D_\phi(x, y) = \|x - y\|_2^2$ if $\phi(x) = \|x\|_2^2$,
- Itakura-Saito divergence: $D_\phi(x, y) = \frac{x}{y} - \log(\frac{x}{y}) - 1$ if $\phi(x) = -\log x$,
- KL divergence: $D_\phi(x, y) = x^\top \log(\frac{x}{y})$ if $\phi(x) = x^\top \log(x)$, and
- Mahalanobis divergence: $D_\phi(x, y) = (x - y)^\top A(x - y)$ if $\phi(x) = x^\top Ax$ and $A \succeq 0$.

As a divergence function, $D_\phi(x, y)$ is always nonnegative by the convexity of ϕ . $D_\phi(x, y) = 0$ if and only if $x = y$. However, it may not be a metric because it may not be symmetric in general, and it may violate the triangle inequality. The asymmetry of Bregman divergence can potentially be more desirable in the setting of comparing distributions than a symmetric measure such as L_p -Wasserstein distance. In Pardo and Vajda (2003), they show an asymptotic equivalence between f -divergences (in particular, χ^2 -divergence) and Bregman divergences under some conditions.

k -means clustering Using Bregman. In Banerjee, Guo, and H. Wang (2005), they show that conditional expectation is the optimal predictor for all Bregman divergences. Moreover, Bregman divergences are the only class of such loss functions. Mathematically,

Theorem 2.1.1. (Theorem 4 in Banerjee, Guo, and H. Wang (2005)) Let $F : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$ be a nonnegative function such that $F(x, x) = 0, \forall x \in \mathbb{R}^d$. Assume that $F(x, y)$ and $F_{x_i, x_j}(x, y), 1 \leq i, j \leq d$ are all continuous. For all random variables X taking value in \mathbb{R}^d , if $\mathbb{E}[X|\mathcal{G}]$ is the unique minimizer of $\mathbb{E}[F(X, Y)]$ overall random variable $Y \in \mathcal{G}$, i.e.

$$\arg \min_{y \in \mathcal{G}} \mathbb{E}[F(X, Y)] = \mathbb{E}[X|\mathcal{G}]$$

where \mathcal{G} is a σ -algebra, then $F(x, y) = D_\phi(x, y)$ for some strictly convex and differentiable function $\phi : \mathbb{R}^d \rightarrow \mathbb{R}$.

This property also ensures the convergence of k -means algorithm in statistical learning when Bregman divergence is used as a loss function. See Banerjee, Merugu, et al. (2005) for more details.

Bregman Projection. In Bauschke and Borwein (1996), they introduced a related class called the Bregman-Legendre functions and the following theorem. Define the Bregman projection of y onto a region Ω as

$$P_{\Omega}(y) = \arg \min_{\omega \in \Omega} D_{\phi}(\omega, y),$$

then the Bregman version of the Pythagorean theorem states that

$$D_{\phi}(x, y) \geq D_{\phi}(x, P_{\Omega}(y)) + D_{\phi}(P_{\Omega}(y), y).$$

Connections with Exponential Family. In Banerjee, Merugu, et al. (2005), they show that there is a one-to-one correspondence between Bregman divergences and exponential families. That is, take an exponential family in a canonical form of:

$$p_{\theta}(x) = \exp(\theta^T x - \psi(\theta))h(x),$$

where $\theta, x \in \mathbb{R}^d$. ψ is the cumulant function with its Legendre convex conjugate ϕ defined as

$$\phi(x) = \sup_t [\langle x, t \rangle - \psi(t)].$$

Then

$$p_{\theta}(x) = \exp(-D_{\phi}(x, \mu(\theta)) - g_{\phi}(x)),$$

with $\mu(\theta) = \nabla \psi(\theta)$. This one-to-one correspondence comes from the duality property of Bregman divergence, which states that

$$D_{\phi}(p, q) = D_{\psi}(q^*, p^*),$$

with $p^* = \nabla \phi(p)$ and $q^* = \nabla \phi(q)$.

Bias-Variance Decomposition. In Buja, Stuetzle, and Shen (2005), they show that the expected Bregman divergence has a bias-variance decomposition

$$\mathbb{E}D_{\phi}(\hat{\theta}, \theta) = D_{\phi}(\mathbb{E}\hat{\theta}, \theta) + \mathbb{E}D_{\phi}(\hat{\theta}, \mathbb{E}\hat{\theta}).$$

Setting $\phi(x) = \|x\|_2^2$ recovers the usual bias-variance decomposition for squared-error loss,

$$\begin{aligned} \mathbb{E}D_{\phi}(\hat{\theta}, \theta) &= \mathbb{E}[(\hat{\theta} - \theta)^2] \\ &= (\mathbb{E}\hat{\theta} - \theta)^2 + \mathbb{E}[(\hat{\theta} - \theta)^2] \\ &= D_{\phi}(\mathbb{E}\hat{\theta}, \theta) + \mathbb{E}D_{\phi}(\hat{\theta}, \mathbb{E}\hat{\theta}). \end{aligned}$$

Figure 2.1 shows how various choices of loss functions can lead to different measures of bias-variance tradeoff in selecting the number of neighbors for the k -nearest neighbor (k -NN) algorithm. For each plot, the solid dot indicates the parameter that minimizes the corresponding loss function. The data used for this illustration come from the `spam` dataset collected at Hewlett-Packard Labs, readily available in the R package `kernelab`.

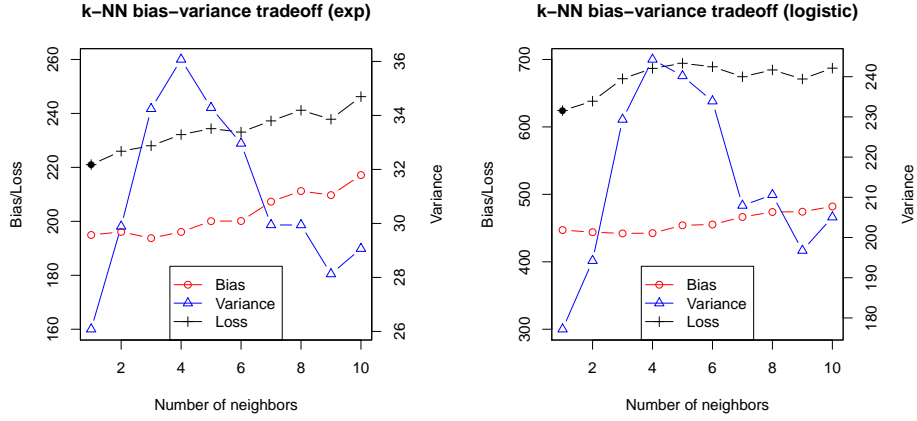


Figure 2.1: The bias-variance tradeoff of the k -NN algorithm based on the loss functions $D_\phi(x, y) = e^x - e^y - e^y(x - y)$ and $D_\phi(x, y) = x \log\left(\frac{x}{y}\right) + (1 - x) \log\left(\frac{1-x}{1-y}\right)$.

In addition, the following lemma will be useful for our analysis later.

Lemma 2.1.2. *Assume that $\phi : \mathcal{X} \rightarrow \mathbb{R}$ is a strictly convex and twice-differentiable function with an L -Lipschitz continuous gradient,*

$$D_\phi(x, y) \leq \frac{L}{2} \|x - y\|_2^2$$

for any $x, y \in \mathcal{X} \subset \mathbb{R}^d$.

Proof. This is clear,

$$\begin{aligned} D_\phi(x, y) &= \phi(x) - \phi(y) - \langle \nabla \phi(y), x - y \rangle \\ &= \int_0^1 \langle \nabla \phi(tx + (1-t)y), x - y \rangle dt - \langle \nabla \phi(y), x - y \rangle \\ &= \int_0^1 \langle \nabla \phi(tx + (1-t)y) - \nabla \phi(y), x - y \rangle dt \\ &\leq \left(\int_0^1 t dt \right) L \|x - y\|_2^2 = \frac{L}{2} \|x - y\|_2^2. \end{aligned}$$

where the second equality comes from the mean value theorem and the inequality comes from the fact that ϕ is a twice-differentiable function with an L -Lipschitz continuous gradient. \square

2.2 Wasserstein Distance

Wasserstein distance is a divergence defined between probability distributions on a given metric space. It is also known as Kantorovich-Monge-Rubinstein metric. The concept is introduced independently in a series of literatures including Gini (1914), Kantorovitch (1958), Kantorovich and Rubinstein (1958), Vaserstein (1969), Mallows et al. (1972), and Tanaka (1973). Wasserstein distance has deep connections with the optimal transport problem, which is also called the Monge-Kantorovich problem, and is discussed first in Monge (1781) and gains a linear programming formulation in Kantorovitch (1958).

Notations

Throughout this section and Chapter 3, the following notations are used unless otherwise stated.

If $x \in \mathbb{R}^d$ denotes a vector in Euclidean space and X represents a matrix, then x^\top denotes the transpose of this vector x , $\|x\|_q$ denotes that q -norm of x , and $\log(x)$ denotes the component-wise logarithm of this vector x . $X \succeq 0$ or $\succ 0$ means that X is positive semi-definite or positive definite, respectively. $\mathcal{X} \subset \mathbb{R}^d$ denotes a set where the diameter of \mathcal{X} is defined as

$$\text{diam}(\mathcal{X}) = \max_{x_1, x_2 \in \mathcal{X}} \|x_1 - x_2\|_2,$$

and $1_{\mathcal{X}}$ denotes an indicator function of the set \mathcal{X} . If \mathbb{P} and \mathbb{Q} are two probability distributions, $\mathcal{P}(\mathcal{X})$ denotes the set of probability distributions defined on \mathcal{X} , then $\Pi(\mathbb{P}, \mathbb{Q})$ denotes the set of all couplings of \mathbb{P} and \mathbb{Q} , i.e. the set of all joint distributions over $\mathcal{X} \times \mathcal{X}$ with marginal distributions being \mathbb{P} and \mathbb{Q} . We use ϕ for a strictly convex and twice-differentiable function with an L -Lipschitz continuous gradient, i.e.

$$0 \prec \nabla^2 \phi(x) \preceq LI_d,$$

where $x \in \text{dom}(\phi)$, i.e. the domain of ϕ , and I_d is an identity matrix in $\mathbb{R}^{d \times d}$. For the statistical learning setup, we define \mathbb{P}_r as an unknown true probability distribution, \mathbb{P}_n as the empirical distribution based on n observations from \mathbb{P}_r , and $\{\mathbb{P}_\theta : \theta \in \mathbb{R}^d\}$ as a parametric family of probability distributions.

Definition 2.2.1. *The Wasserstein distance of order p between the probability distributions \mathbb{P} and \mathbb{Q} is defined as*

$$W_p(\mathbb{P}, \mathbb{Q}) = \left(\inf_{\pi \in \Pi(\mathbb{P}, \mathbb{Q})} \int_{\mathcal{X} \times \mathcal{X}} [c(x, y)]^p \pi(dx, dy) \right)^{1/p}, \quad (2.1)$$

where $p \geq 1$. $c(\cdot, \cdot) \geq 0$ is a metric supported on $\mathcal{X} \times \mathcal{X}$. An important special case is the Wasserstein- L^q distance of order p as follows,

$$W_p^{L^q}(\mathbb{P}, \mathbb{Q}) = \left(\inf_{\pi \in \Pi(\mathbb{P}, \mathbb{Q})} \int_{\mathcal{X} \times \mathcal{X}} \|x - y\|_q^p \pi(dx, dy) \right)^{1/p}. \quad (2.2)$$

$q = 2$ and $\mathcal{X} = \mathbb{R}^d$ in (2.2) corresponds to the squared Wasserstein- L^2 distance of order 2:

$$W_2^{L^2}(\mathbb{P}, \mathbb{Q}) = \left(\inf_{\pi \in \Pi(\mathbb{P}, \mathbb{Q})} \int_{\mathcal{X} \times \mathcal{X}} \|x - y\|_2^2 \pi(dx, dy) \right)^{1/2} \quad (2.3)$$

Remark 1. For probability distributions, convergence under Wasserstein distance of order p is equivalent to weak convergence plus convergence of the first p moments (see Theorem 6.8 in Villani (2008) for details). Given \mathbb{P} and \mathbb{Q} , we have the following two properties of the Wasserstein distance of order p ,

1. $W_p(\mathbb{P}, \mathbb{Q}) \geq 0$ and the equality holds if and only if $\mathbb{P} = \mathbb{Q}$ almost everywhere.
2. $W_p(\mathbb{P}, \mathbb{Q})$ is a metric since $W_p(\mathbb{P}, \mathbb{Q}) = W_p(\mathbb{Q}, \mathbb{P})$ and

$$W_p(\mathbb{P}, \mathbb{Q}) \leq W_p(\mathbb{P}, \mathbb{S}) + W_p(\mathbb{S}, \mathbb{Q}),$$

where \mathbb{S} is another probability distribution.

The form of Equation 2.1 also provides a nice interpretation. Let $c(x, y)$ be the cost function in the definition of $W_p(\mathbb{P}, \mathbb{Q})$. As shown in Figure 2.2, $c(x, y)\pi(dx, dy)$ represents the cost of transferring the probability density of \mathbb{P} at x to probability density of \mathbb{Q} at y . The minimization over all coupling distributions corresponds to finding the policy that minimizes the cost of transforming one probability distribution to the other.

Applications

Wasserstein distances are commonly used in optimal transport (Villani 2008), and also have applications in many other areas, such as the study of mixing for Markov chains in probability theory (Dobrushin 1996; Peres 2005), rates of fluctuations for empirical measures in statistics (Rachev 1991; Rachev and Ruschendorf 1998; Dobrić and Yukich 1995), and propagation of chaos in statistical mechanics (Dobrushin 1970; Spohn 2012).

The Wasserstein- L^2 distance, also known as Mallows distance, has been widely used in topics such as statistical testing (Munk and Czado 1998; De Wet 2002), machine learning (D. Zhou, Jia Li, and Zha 2005), and stochastic games (Lasry and Lions 2007).

Distribution Learning with L_1 -Wasserstein Distance. In Arjovsky, Chintala, and Bottou (2017), they use neural networks to learn probability density and define the objective function for optimization to be the L_1 -Wasserstein distance. They have shown promising results on a numerical experiment in image generation. We discuss this in detail in Section 2.4.

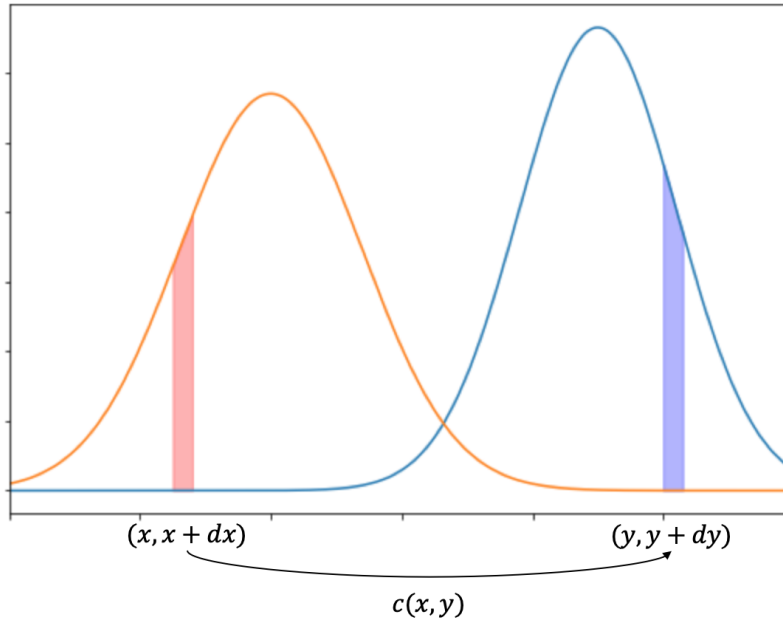


Figure 2.2: Earthmover interpretation of Wasserstein distance

2.3 Neural Networks

Artificial neural networks are a class of models that commonly consist of stacks of layers of non-linear transforms. The basic building blocks of neural networks are neurons, which apply a non-linear transformation (called activation) to the real number that is fed as input. A layer is an aggregation of neurons. The most common layer is called a fully-connected layer and has the mathematical form $y = f(Wx + b)$. Here $x \in \mathbb{R}^{m \times n}$ is the input of the layer and $y \in \mathbb{R}^{k \times n}$ is the output. $W \in \mathbb{R}^{k \times m}$ is a matrix to perform linear transform and $b \in \mathbb{R}^{k \times n}$ is the bias term. $f : \mathbb{R}^{k \times n} \rightarrow \mathbb{R}^{k \times n}$ is called the activation function, which could be one of the following popular choices:

- ReLU activation: $f(t) = \max(0, t)$, element-wise,
- Sigmoid activation: $f(t) = \frac{1}{1 + \exp(-t)}$, element-wise,
- Tanh activation: $f(t) = \tanh(t)$, element-wise,
- Softmax activation: $f(t)_i = \frac{\exp(t_i)}{\sum \exp(t_i)}$, i is the index of every entry of t .

Neural networks are, in essence, function approximators. They are trained by minimizing a task-dependent loss function, through stochastic gradient descent (or its variants). The gradient is calculated using chain-rule iteratively with respect to all trainable parameters in

the network. This method is also called back-propagation because the parameters in later layers are closer to the outputs and the gradients of them are hence computed first.

As a simple example, consider a two-layer neural network with a single neuron in the hidden (middle) layer and all the input and outputs are one-dimensional. Its outputs can be expressed as $y_1 = f(W_1x + b_1)$ and $y_2 = f(W_2y_1 + b_2)$. Here W_i , b_i and y_i are the weight, bias, and output of the i th layer, $i = 1, 2$. Then the gradient of the final output y_2 with respect to W_2 is

$$\frac{\partial y_2}{\partial W_2} = y_1 f'(W_2 y_1 + b_2),$$

and the gradient with respect to W_1 can be written as

$$\frac{\partial y_2}{\partial W_1} = \frac{\partial y_2}{\partial y_1} \frac{\partial y_1}{\partial W_1} = W_2 f'(W_2 y_1 + b_2) \cdot x f'(W_1 x + b_1).$$

Here the first equality is using the chain-rule.

LSTM units Long Short-Term Memory is a type of neural network structure. The concept was introduced by Hochreiter and Schmidhuber (1997). Gers, Schmidhuber, and Cummins (1999) improve LSTM by adding a forget gate. Figure 2.3 describes the structure of a typical LSTM unit. Let $h \in \mathbb{R}$ be the dimension of the hidden space. The vector $c_t \in \mathbb{R}^h$ is called a cell state vector, which memorizes the state of the unit at index t . The vector $x_t \in \mathbb{R}^d$ is the input vector of the unit, while the vector $h_t \in \mathbb{R}^h$ is the output vector of the unit. In the graph, the nodes with circles denote element-wise operations (multiplications \times and additions $+$), while the nodes with rectangles denote layers with sigmoid or tanh activations. The edges represent the variables.

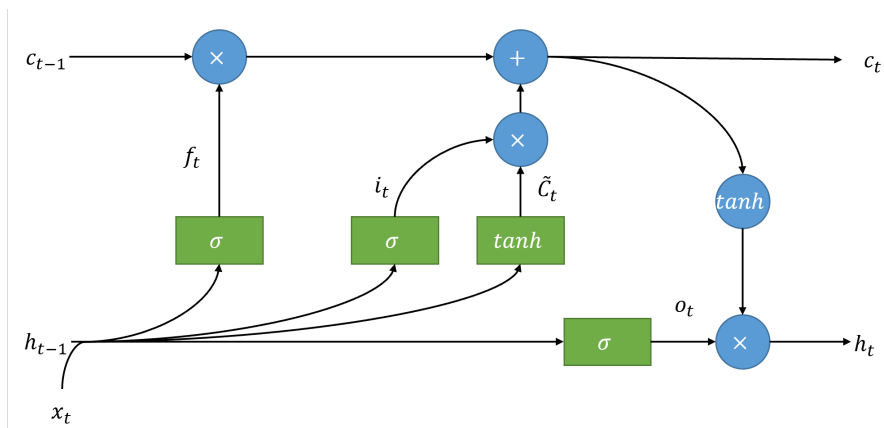


Figure 2.3: Structure of an LSTM unit

Inside each unit, f_t is called the forget gate's activation vector, i_t is called the input gate's activation vector, o_t is called the output gate's activation vector, and \tilde{C}_t is called the

candidate state vector. These interactions among variables and operations can be described with the following equations:

$$\begin{aligned} i_t &= \sigma(x_t U_i + h_{t-1} W_i + b_i) \\ f_t &= \sigma(x_t U_f + h_{t-1} W_f + b_f) \\ o_t &= \sigma(x_t U_o + h_{t-1} W_o + b_o) \\ \tilde{C}_t &= \tanh(x_t U_g + h_{t-1} W_g) \\ c_t &= \sigma(f_t \circ c_{t-1} + i_t \circ \tilde{C}_t) \\ h_t &= \tanh(c_t \circ o_t) \end{aligned}$$

Here \circ represents the element-wise product. W 's and U 's are the weights of each layer and b 's are the biases of each layer.

These layers learn to put weights on inputs from the current t and outputs from previous outputs. With these vectors and operations, LSTM can model sequential data by remembering and forgetting information from previous units, hence learning long-term dependency among inputs. We use LSTM in Section 5.3 to model a sequence of Credit Default Swap Index orders.

2.4 Generative Adversarial Networks

Generative Adversarial Networks (GANs) (Goodfellow et al. 2014) provide a versatile class of models for generative modeling. The key idea behind GANs is to interpret the process of generative modeling as a competing game between two neural networks: a generator network G and a discriminator network D . The generator network G attempts to fool the discriminator network by converting random noise into sample data or, equivalently, transforming an input distribution into approximately the distribution underlying the dataset, while the discriminator network D is a classifier that tries to identify whether the input sample is a fake data sample or a true data sample.

Applications of GANs

Since their introduction to the machine learning community, the popularity of GANs have grown exponentially with numerous applications. Examples include high resolution image generation (Denton, Chintala, and Fergus 2015; Radford, Metz, and Chintala 2015; C. Li and Wand 2016; Zhang, Goodfellow, et al. 2018), image inpainting (Yeh et al. 2017), image super-resolution (Ledig et al. 2017), visual manipulation (J.-Y. Zhu, Krähenbühl, et al. 2016), text-to-image synthesis (Reed et al. 2016; Zhang, Xu, et al. 2017), video generation (Vondrick, Pirsiaavash, and Torralba 2016), semantic segmentation (Denton, Chintala, and Fergus 2015), abstract reasoning diagram generation (Kulharia et al. 2017), style transfer and domain adaptation (Bousmalis et al. 2017; Isola et al. 2017; J.-Y. Zhu, Park, et al. 2017; Antipov,

Baccouche, and Dugelay 2017) and noise reduction (Wolterink et al. 2017). See also Arjovsky (2017), Jerry Li et al. (2017), and Mescheder, Nowozin, and Geiger (2017) for more details on the training dynamics of GANs.

Advantages of using GANs

First of all, the generator network has so many parameters that it is close to being nonparametric. In most cases, it does not depend on any assumptions to build, train, or implement, nor does it rely on rich knowledge about prior distributions. Therefore, it is suitable for tasks where the underlying dynamic or distribution is unclear, or where a simple model would result in a over-simplified solution.

Second, the model capacity of GANs can be arbitrarily large so long as the network is deep enough. It is hence capable of learning complex underlying distributions, especially in tasks related to computer vision.

Third, during training the generator faces an ever-changing loss function. This has benefits in a number of situations. Sometimes there is a dilemma choosing between L^1 and L^2 losses and one is unwilling to use either of them. Sometimes there are hardly any pre-determined loss functions to optimize. For example, how could one tell the quality of generated image samples? Sometimes the samples are unpaired, so there is no baseline for comparison. The discriminator adds flexibility to these situations because it is improved during each step of training and acts as an adversary to the currently best generator.

Lastly, GANs can utilize ideas and structures from the huge and fast-growing model zoos contributed by the deep learning community. For instance, pretrained classifiers such as the Inception Network can be used as discriminators without much modification in some applications. Encoded messages from autoencoders can be directly used as inputs to GANs. Besides, convolution and transpose convolutions can be incorporated into the generator network. Attention mechanics LSTM units can improve the performance for generating sequences like sentences or music.

Basics of GANs: an example on MNIST

To start with a simple example, consider how images could be generated that look like the hand-written digits from the MNIST dataset.

Network structure. In this dataset, each sample is a $28 * 28$ (784 pixels in total) image that contains a black-and-white hand-written 0-9 digit. In this case, the generator can take a low-dimensional random noise $z \in \mathbb{R}^{100}$. z first goes through a fully-connected layer with ReLU as activation function and its dimension is raised to 128. It then goes through another fully-connected layer with tanh as activation function and its dimension is raised to 784. The output is the sample generated by the generator.

The discriminator, on the other hand, takes a 784-dimensional vector as input, which can either be a true image from the dataset or a fake image generated by the generator. The

input then goes through one hidden layer with Maxout as the activation function. After going through a sigmoid layer, the final output is between 0 and 1, which represents the confidence that the discriminator thinks the input is a true image. Refer to Figure 2.4 for the structure of this vanilla GAN on MNIST.

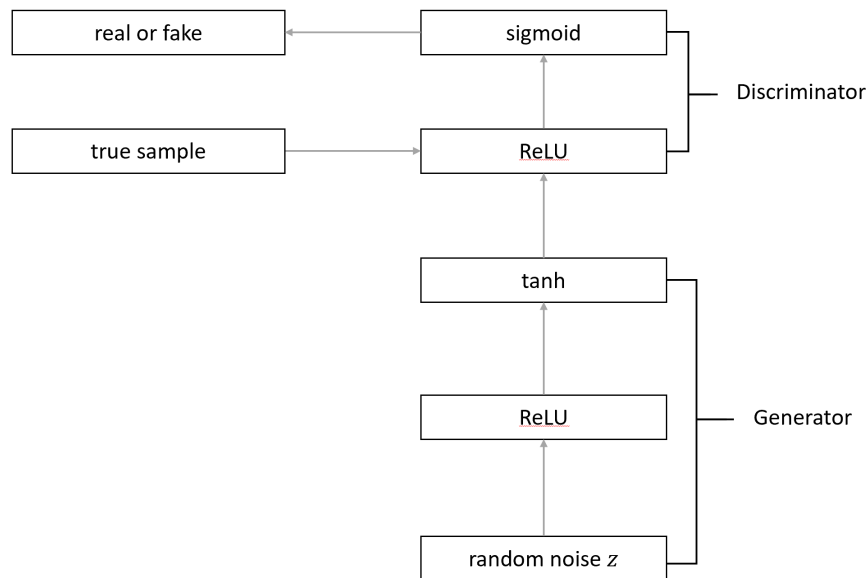


Figure 2.4: Network structure of the vanilla GAN

Training of GANs. Training of GANs is often considered to be difficult. Careful hyperparameter tuning is required as otherwise the generator would only learn to replicate the samples from the discriminator, or the discriminator is over-powered such that the output of $D(G(x))$ is always 0 for a fake sample x , therefore the gradient no longer carries information, and the training of generator is stuck forever. Many tricks have been developed to address these problems. For example, one may apply a special scheme of decreasing the learning rate or change the ratio of number of iterations the generator and discriminator update themselves through stochastic gradient descent.

Mathematics behind GANs

GANs and Jensen-Shannon Divergence

The goal of the GANs is to estimate a probability distribution \mathbb{P}_r hidden in the data. As defined in Definition 3.2.3 in Chapter 3, one can define a random variable Z with a fixed distribution \mathbb{P}_Z and pass it through a parametric function $g_\theta : \mathcal{Z} \rightarrow \mathcal{X}$ to construct a probability distribution \mathbb{P}_θ . In practice, the parametric function g_θ is implemented using a

neural network called *Generator* G . Meanwhile, another neural network *Discriminator* D will assign a score between 0 to 1 to the generated samples, either from the empirical distribution \mathbb{P}_r or the approximate distribution $\mathbb{P}_\theta = g_\theta(Z)$. A higher score from the discriminator D would indicate that the sample is more likely to be from the empirical distribution. A GAN is trained by optimizing G and D iteratively until D can no longer distinguish between samples from \mathbb{P}_r or \mathbb{P}_θ . In this light, one can learn the probability distribution \mathbb{P}_r by adapting θ and fitting the data with \mathbb{P}_θ . This approximation is done by finding a solution f that optimizes a given cost function between \mathbb{P}_r and \mathbb{P}_θ .

Mathematically, training of GANs with an optimal discriminator is minimizing the Jensen-Shannon divergence between \mathbb{P}_r and \mathbb{P}_θ . Indeed, recall that GANs is a min-max game of

$$\min_G \max_D \left\{ \mathbb{E}_{x \sim \mathbb{P}_r} [\log D(x)] + \mathbb{E}_{z \sim \mathbb{P}(z)} [\log(1 - D(G(z)))] \right\}. \quad (2.4)$$

If we fix G and optimize for D , the optimal discriminator would be $D_G^*(x) = \frac{p_r(x)}{p_r(x) + p_g(x)}$, where p_r and p_g are density functions of \mathbb{P}_r and $\mathbb{P}_\theta = g_\theta(Z)$ respectively. Plugging this back to Equation (2.4),

$$\min_G \left\{ \mathbb{E}_{x \sim \mathbb{P}_r} \left[\log \frac{p_r(x)}{p_r(x) + p_g(x)} \right] + \mathbb{E}_{x \sim \mathbb{P}_\theta(Z)} \left[\log \frac{p_g(x)}{p_r(x) + p_g(x)} \right] \right\} \quad (2.5)$$

$$= -\log 4 + 2JS(\mathbb{P}_r, \mathbb{P}_\theta), \quad (2.6)$$

where the last term is the Jensen-Shannon (JS) divergence.

In Arjovsky, Chintala, and Bottou (2017), the JS divergence is replaced with Wasserstein distance. In Section 3.3, we replace the JS divergence with RW divergence and show that it would result in better performance.

Variants of GANs

There are many variants of GANs. Least square GANs (LSGANs) (Mao et al. 2017) attain a stable performance during the learning process, replacing the sigmoid cross entropy loss by the least square loss in the discriminator network of the original GANs. They are also shown to generate higher quality images than the original GANs in practice. DRAGANs (Kodali et al. 2017) alleviate the instability of the GANs training and offer a clear game-theoretic justification by introducing regret minimization to reach the equilibrium in games and to further explain the reason for the success of simultaneous gradient descent in GANs. Conditional GANs (CGANs) (Mirza and Osindero 2014) propose to stabilize the training by imposing the control on modes of the generated data in an original generative model. Information-theoretic GANs (InfoGANs) (X. Chen et al. 2016) provide highly semantic and meaningful hidden representations on a number of image datasets by maximizing the mutual information between a fixed small subset of GAN's noises and the observations. Auxiliary Classifier GANs (ACGANs) (Odena, Olah, and Shlens 2017) improve GANs by adding more

structure to the latent space together with a specialized cost function and with high-quality samples. They also lead to a new analysis for assessing the discriminability and diversity of samples from class-conditional image synthesis models. Energy-Based GANs (EBGANs) (Zhao, Mathieu, and LeCun 2016) propose a new energy perspective of GANs. They construct an energy function to measure the discriminator that attributes lower energies to the regions near the data manifold and higher energies to other regions. As a result, the EBGAN framework is shown to generate reasonable high-resolution images without a multi-scale approach. Boundary Equilibrium GANs (BEGANs) (Berthelot, Schumm, and Metz 2017) adopt a new equilibrium enforcing method paired with the Wasserstein divergence to train GANs with an auto-encoder. This approach not only balances the generator network and the discriminator network, but also uncovers a novel approximate convergence measure, leading to a fast and stable training with high visual-quality.

Next, we present in detail two of the most important structures of GANs to show how the vanilla GAN’s structure could be improved in various directions best suited for different tasks.

Conditional GAN In the vanilla GAN, the only input is the random noise z .

$$\min_G \max_D \left\{ \mathbb{E}_{x \sim \mathbb{P}_r} [\log D(x)] + \mathbb{E}_{z \sim \mathbb{P}(z)} [\log(1 - D(G(z)))] \right\}. \quad (2.7)$$

In this case, there is no control over what kind of image to output. For example, in the MNIST dataset, the generated image randomly chooses a digit from 0 to 9 as the output. This restricts the use of GANs.

A special GAN structure called Conditional GAN (CGAN) (Mirza and Osindero 2014) solves this issue by adding conditioning to the objective function. That is, in CGAN, they turn to optimize the following objective:

$$\min_G \max_D \left\{ \mathbb{E}_{x \sim \mathbb{P}_r} [\log D(x|y)] + \mathbb{E}_{z \sim \mathbb{P}(z)} [\log(1 - D(G(z|y)))] \right\}. \quad (2.8)$$

How should one implement conditioning in practice? Of course, one way is to build a generator for each individual category (then we would have 10 generators for the MNIST dataset). The clever way is to include the category information into the input. For instance, one could choose to use the structure shown in Figure 2.5 on the MNIST dataset. The categorical information y here is concatenated with the random noise z as input to the generator network.

CycleGAN Previously, the output of GANs are random: the generator takes a random noise and then transforms its distribution to something complex. GANs can do more. For example, style transfer, where the input is no longer random but rather a deterministic and meaningful image. Suppose that dataset X contains photos of various landscapes and dataset Y contains paintings from one’s favourite painter. The samples in the dataset X and Y are of course unpaired. Now we wish to transform the photos in a way such that they gain the styles from the paintings. This type of task is called style transfer.

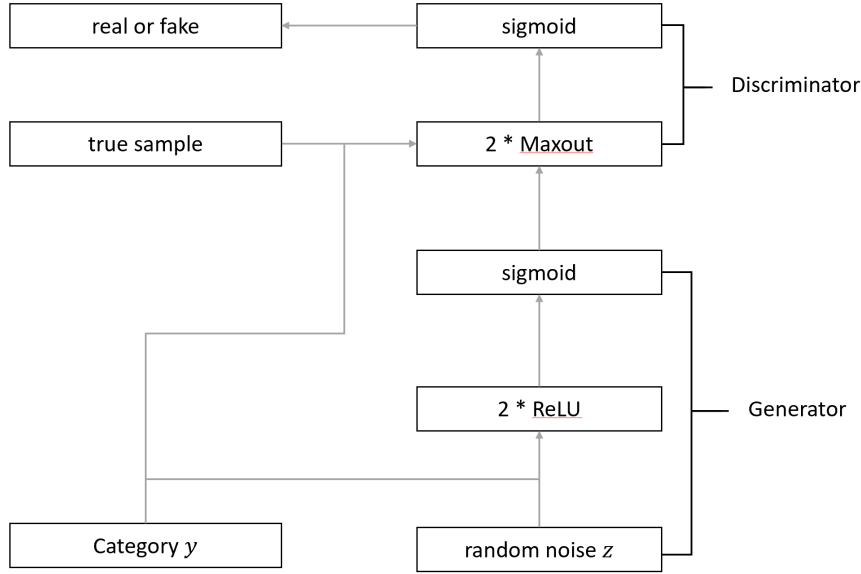


Figure 2.5: Network structure of the conditional GAN

To do this, we introduce a special structure called CycleGAN (J.-Y. Zhu, Park, et al. 2017). It consists of two pairs of networks: (G, D_X) that takes input from dataset X and (F, D_Y) that takes input from dataset Y . G and F are generators that transform the input sample to the dataset X and Y , respectively. D_X and D_Y are discriminators that identify whether a sample is from dataset X and Y or not, respectively.

The loss function now contains three parts: the adversarial loss for G and D_Y ,

$$\mathcal{L}_G = \mathbb{E}_{y \sim p_{\text{data}}(y)}[\log D_Y(y)] + \mathbb{E}_{x \sim p_{\text{data}}(x)}[\log(1 - D_Y(G(x)))], \quad (2.9)$$

the adversarial loss for F and D_X ,

$$\mathcal{L}_F = \mathbb{E}_{x \sim p_{\text{data}}(x)}[\log D_X(x)] + \mathbb{E}_{y \sim p_{\text{data}}(y)}[\log(1 - D_X(F(y)))], \quad (2.10)$$

and the cycle consistency loss that connects both,

$$\mathcal{L}_{\text{cyc}}(G, F) = \mathbb{E}_{x \sim p_{\text{data}}(x)}[\|F(G(x)) - x\|_1] + \mathbb{E}_{y \sim p_{\text{data}}(y)}[\|F(G(y)) - y\|_1]. \quad (2.11)$$

The objective function is a weighted sum of all three losses:

$$\mathcal{L} = \mathcal{L}_G + \mathcal{L}_F + \lambda \mathcal{L}_{\text{cyc}}(G, F).$$

Multiple tricks are applied to enhance the training process. For example, to stabilize training, CycleGAN uses L^2 loss instead of the Jensen-Shannon divergence. In particular, we train G to minimize

$$\mathbb{E}_{x \sim p_{\text{data}}(x)}[(D(G(x)) - 1)^2],$$

and train D to minimize

$$\mathbb{E}_{x \sim p_{\text{data}}(x)}[D(G(x))^2] + \mathbb{E}_{y \sim p_{\text{data}}(y)}[(D(y) - 1)^2].$$

The discriminator D is updated based on a history 'momentum' of generated images rather than the most recent ones. D is also designed in a fully-convoluted way to reduce the number of trainable parameters and to make it work for inputs of any size.

Wasserstein GANs

A recurring theme to improve GANs training is the choice of loss functions (divergence functions). The first proposed class of loss functions is based on the Jensen-Shannon (JS) divergence, which is essentially the symmetric version of the Kullback-Leibler (KL) divergence. It is shown in Arjovsky, Chintala, and Bottou (2017) that JS divergence is undesirable with unstable training, suggesting Wasserstein- L^1 distance as an alternative. The resulting Wasserstein GANs (WGANs) outperform the original GANs in several aspects. The Wasserstein- L^1 distance is continuous, differentiable, and has a duality representation. This constant availability of derivatives ensures a stable gradient descent algorithm in the process of training. Besides the stability, the Wasserstein- L^1 distance also avoids the issue of mode collapse. Mode collapse is a phenomenon that the generator learns only the modes of the training set. This is caused by training the generator against a fixed discriminator (Goodfellow et al. 2014). The ability of WGANs to train the discriminator to optimality solves this problem. WGANs further provide meaningful learning curves that can be used for debugging and for hyperparameter searching. The volatility of the gradients is controlled with additional regularization techniques such as weight clipping (Arjovsky, Chintala, and Bottou 2017) and gradient penalty (Gulrajani et al. 2017).

The algorithm of WGAN is summarized in Algorithm 1. Line 7 is the weight clipping step. It enforces the Lipschitz continuity of the critic (discriminator). The details are discussed in Chapter 3.

If instead of weight clipping we apply a penalty to the gradient, we will result in Algorithm 2, which is proposed in Gulrajani et al. (2017) and is called WGAN with gradient penalty (WGAN-GP). The gradient penalty term is in Line 7.

2.5 Variational Autoencoders

Variational autoencoders (VAEs) (Kingma and Welling 2013; Rezende, Mohamed, and Wierstra 2014) are another type of generative model that consist of two neural networks: an encoder network and a decoder network $P_{\theta}(X|z)$. The encoder network encodes a sample X into parameters of a probability distribution $Q_{\lambda}(z|X)$. A latent representation of the input, or the encoded message, z , is sampled according to $Q_{\lambda}(z|X)$. The decoder network finally decodes z back into X' , a reconstructed version of X . The loss function of the variational autoencoder ensures that X' will be close to X after convergence. Similar to other

Algorithm 1 WGAN. The default values $\alpha = 0.00005$, $c = 0.01$, $m = 64$, $n_{critic} = 5$.

Require: α : the learning rate; c : the clipping parameter; m : the batch size; n_{critic} , the number of iterations of the critic per generator iteration.

Require: w_0 , initial critic parameters; θ_0 : initial generator's parameters.

```

1: while  $\theta$  has not converged do
2:   for  $t = 0, \dots, n_{critic}$  do
3:     Sample a batch of real data  $\{x_i\}_{i=1}^m$  from  $\mathbb{P}_r$ .
4:     Sample a batch of prior samples  $\{z_i\}_{i=1}^m$  from  $p(z)$ .
5:      $g_w \leftarrow \frac{1}{m} \sum_{i=1}^m [\nabla_w f_w(x_i) - \nabla_w f_w(g_\theta(z_i))]$ .
6:      $w \leftarrow w + \alpha \cdot \text{RMSPProp}(w, g_w)$ .
7:      $w \leftarrow \text{clip}(w, -c, c)$ .
8:   end for
9:   Sample a batch of prior samples  $\{z_i\}_{i=1}^m$  from  $p(z)$ .
10:   $g_\theta \leftarrow -\frac{1}{m} \sum_{i=1}^m \nabla_{\theta} f_w(g_\theta(z_i))$ .
11:   $\theta \leftarrow \theta - \alpha \cdot \text{RMSPProp}(\theta, g_\theta)$ .
12: end while

```

encoding-decoding structures, the encoder learns a low-dimensional representation of the input X , which can be used as a non-linear dimension-reduction technique. The decoder, on the other hand, takes any random input z from a simple distribution and yields a synthetic sample. The generated sample is unlikely to be in the original dataset since the decoder is a continuous mapping. Therefore, the decoder network can be used as a simulator.

Mathematics behind Variational Autoencoders

Consider a latent variable z and a sample X taken from a dataset. As a generative model, the objective of a VAE is to maximize the probability that the sample X appears in the synthetic dataset given we use all possible latent configurations, that is, we want to maximize:

$$P(X) = \int P(X|z; \theta)P(z)dz, \quad (2.12)$$

where $P(z)$ is the prior distribution of the latent variable, $P(X|z; \theta)$ is the likelihood function, parametrized by θ . However, since this integral is very hard to evaluate, observe that by the definition of KL divergence,

$$\begin{aligned} D_{\text{KL}}(Q(z|X), P(z|X)) &= \mathbb{E}_{z \sim Q}[\log Q(z|X) - \log P(z|X)] \\ &= \mathbb{E}_{z \sim Q}[\log Q(z|X) - \log P(X|z) - \log P(z)] + \log P(X) \\ &= \mathbb{E}_{z \sim Q}[\log P(X|z)] - D_{\text{KL}}(Q(z|X), P(z)) + \log P(X), \end{aligned}$$

where the second equality is using the Bayes rule and the third equality is again by the definition of KL divergence. We obtain this relationship:

$$\log P(X) - D_{\text{KL}}(Q(z|X), P(z|X)) = \mathbb{E}_{z \sim Q}[\log P(X|z)] - D_{\text{KL}}(Q(z|X), P(z)). \quad (2.13)$$

Algorithm 2 WGAN-GP. The default values $\alpha = 0.00005$, $\lambda = 10$, $n_{critic} = 5$. $\beta_1 = 0$, $\beta_2 = 0.9$

Require: α, β_1, β_2 : the parameters in the Adam optimizer; m : the batch size; n_{critic} , the number of iterations of the critic per generator iteration, λ : the gradient penalty coefficient.

Require: w_0 , initial critic parameters; θ_0 : initial generator's parameters.

```

1: while  $\theta$  has not converged do
2:   for  $t = 0, \dots, n_{critic}$  do
3:     for  $i = 1, \dots, m$  do
4:       Sample real data  $x$  from  $\mathbb{P}_r$ , latent variable  $z$  from  $p(z)$ , a random number  $\epsilon$  that
         follows uniform[0,1].
5:        $\tilde{x} \leftarrow G_\theta(z)$ 
6:        $\hat{x} \leftarrow \epsilon x + (1 - \epsilon)\tilde{x}$ 
7:        $L^{(i)} \leftarrow D_w(\tilde{x}) - D_w(x) + \lambda(\|\nabla_{\hat{x}} D_w(\hat{x})\|_2 - 1)^2$ 
8:     end for
9:      $w \leftarrow \text{Adam}(\nabla_w \frac{1}{m} \sum_{i=1}^m L^{(i)}, w, \alpha, \beta_1, \beta_2)$ 
10:  end for
11:  Sample a batch of latent samples  $\{z_i\}_{i=1}^m$  from  $p(z)$ .
12:   $\theta \leftarrow \text{Adam}(\nabla_\theta \frac{1}{m} \sum_{i=1}^m -D_w(G_\theta(z)), \theta, \alpha, \beta_1, \beta_2)$ .
13: end while

```

On the left hand side of Equation 2.13, the divergence term can be negligible if the encoding network $Q(z|X)$ has high capacity. The right hand side is called the Evidence Lower Bound (ELBO) and is the actual objective function that the VAE is maximizing. In practice, the prior $P(z)$ is usually taken as the standard Gaussian distribution $N(0, I)$. The log-likelihood term can be treated as the reconstruction loss when a sample X is first encoded into z and then reconstructed back to itself. Under the Gaussian assumption, it becomes the common L^2 loss. Therefore, the loss function for VAE is written as the sum of squared errors plus KL divergence regularization. The whole procedure is visualized in Figure 2.6.

Compared to GANs, VAEs focus more on the relationship between samples and their encodings. The applications of VAEs include anomaly detection (An and Cho 2015), image caption modeling (Pu et al. 2016), discrete data generation (Kusner, Paige, and Hernández-Lobato 2017; Semeniuta, Severyn, and Barth 2017), representation learning (Higgins et al. 2017) and recommendation systems (X. Li and She 2017). We use VAEs to perform metagenomic binning in Section 5.2.

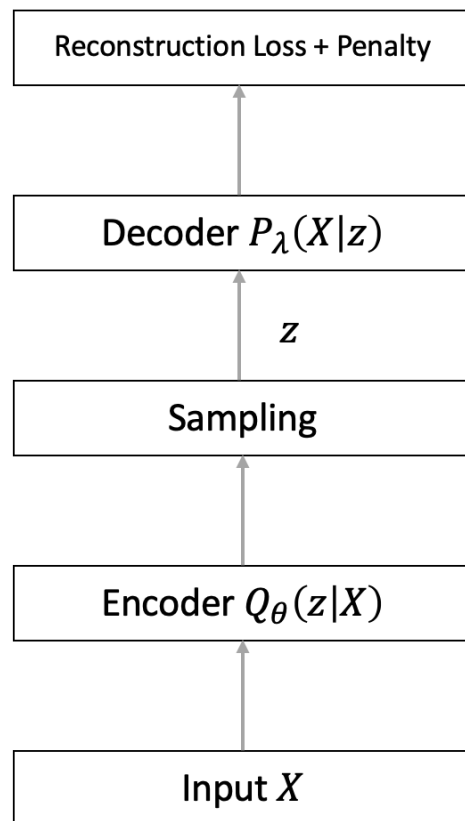


Figure 2.6: Structure of a variational autoencoder

Chapter 3

Relaxed Wasserstein GANs

In this chapter, we introduce a new family of statistical divergences: the Relaxed Wasserstein (RW) divergence. We start by showing that it has strong probabilistic and statistical properties. We then show its competitive performance in building Generative Adversarial Networks.

3.1 Introduction

Statistical divergences play an important role in many data-driven applications. In addition to generative adversarial networks which we introduce in Chapter 2, other striking examples include statistical learning and robust optimization.

In both the literature of learning and robust optimization, one popular choice to measure the difference between two distributions is the Kullback-Leibler divergence, which has strong theoretical foundation in information theory and large deviations (Pardo and Vajda 1997). However, there are two issues in using the KL divergence. The first one is that the KL divergence between a continuous distribution and its empirical version, which is always a discrete distribution, is undefined (or infinite). The second issue is that KL divergence does not take into consideration the relative position of probability mass. As an example, consider the discrete distribution \mathbb{P} which puts 1/2 mass on 0 and 1/2 mass on 1, and the discrete distribution \mathbb{Q} which puts 1/2 mass on ϵ and 1/2 mass on $1 - \epsilon$. The KL divergence does not reflect the convergence of \mathbb{Q} to \mathbb{P} as $\epsilon \downarrow 0$, hence it is too restrictive. It is therefore natural to use alternative measures for distributions, such as f -divergence, L_p -Wasserstein distance, and Prohorov metric.

In this chapter, we propose a novel class of statistical divergence called *Relaxed Wasserstein* (RW) divergence. RW divergence is Wasserstein distance parametrized by the class of strictly convex and differentiable functions, which contain different curvature information. We first show that RW divergence is dominated by the total variation (TV) distance and squared Wasserstein- L^2 divergence (Theorem 3.2.1). In parallel to the Wasserstein- L^2 divergence, we obtain its nonasymptotic moment estimate (Theorem 3.2.2) and its concentration inequality (Theorem 3.2.3). By comparing with Wasserstein divergence, we show RW is a reasonable

divergence.

For application purposes, we establish an important lemma (Lemma 3.2.4) which states that RW divergence can be a distorted Wasserstein- L^2 divergence with some residual terms independent of the coupling. This decomposition immediately leads to the continuity and differentiability of RW divergence (Theorem 3.2.5). From a practical perspective, especially in light of stochastic gradient descent for GANs, these properties ensure the plausibility of a gradient descent procedure. Using the decomposition lemma again, we establish the duality representation of RW divergence (Theorem 3.2.6), which gives rise to an explicit formula for the gradient evaluation and an asymmetric clipping procedure (Corollary 3.2.6.1). Our numerical experiments show that this asymmetric clipping is useful for controlling the volatility of the gradient.

We illustrate the use of RW divergence in GANs. In particular, we introduce Relaxed Wasserstein GANs (RWGANs) and compare RWGANs with several state-of-the-art GANs in image generation. We use RWGANs with KL divergence and the architectures of DCGAN and MLP. We first evaluate all of candidate methods on MNIST and Fashion-MNIST datasets and show that RWGANs are competitive with other popular approaches. Then we conduct the experiment on CIFAR-10 and ImageNet datasets to investigate if RWGANs outperform WGANs with symmetric clipping and gradient penalty, denoted as WGANs and WGANs-GP respectively.

Our numerical results show that despite the fastest rate of training, WGANs-GP fail to converge in some cases; RWGANs are robust and converge faster than WGANs, suggesting that RWGANs strike a balance between WGANs and WGANs-GP and RWGANs might be more desirable for large-scale computations. Furthermore, we observe in our experiments RWGANs are fastest in generating meaningful and diverse images compared to other GANs. In addition, RWGANs attain the highest inception scores at the initial stage of training on the CIFAR-10 dataset, meaning that the generated samples correlate well with human evaluations (Salimans et al. 2016). As a byproduct, our experiment provides some evidences that an appropriate weight clipping has the potential to be competitive with gradient penalty in WGANs.

Section 2.2 already provides the preliminaries and notations that will be used throughout the chapter. The rest of the chapter is organized as follows. Section 3.2 describes the RW divergence and discusses its theoretical properties. Section 3.3 discusses the implementation of RWGANs and presents two numerical studies on four real data examples. We will discuss choices of the convex function parametrizing the RW divergence in Section 4.2 in Chapter 4, and the application of RW divergence to robust optimization problems in Section 5.1 in Chapter 5.

3.2 Relaxed Wasserstein Divergence

We now propose a new class of statistical divergence called Relaxed Wasserstein (RW) divergence, parametrized by Wasserstein divergence and Bregman divergence. The term

relaxed refers to the fact that RW divergence relaxes the symmetry of cost function $c(x, y)$ in Equation (2.1) and extends to a broader class of asymmetric divergences.

Definition 3.2.1. *The Relaxed Wasserstein divergence between the probability distributions \mathbb{P} and \mathbb{Q} is defined as*

$$W_{D_\phi}(\mathbb{P}, \mathbb{Q}) = \inf_{\pi \in \Pi(\mathbb{P}, \mathbb{Q})} \int_{\mathcal{X} \times \mathcal{X}} D_\phi(x, y) \pi(dx, dy),$$

where D_ϕ is the Bregman divergence with a strictly convex and differentiable function $\phi : \mathbb{R}^d \rightarrow \mathbb{R}$.

Remark 2. 1. $W_{D_\phi}(\mathbb{P}, \mathbb{Q}) \geq 0$ and the equality holds if and only if $\mathbb{P} = \mathbb{Q}$ almost everywhere.

2. $W_{D_\phi}(\mathbb{P}, \mathbb{Q})$ is not a metric since $D_\phi(x, y)$ is generally asymmetric.

3. $W_{D_\phi}(\mathbb{P}, \mathbb{Q})$ includes two important special cases, $W_2^{L^2}$ and W_{KL} . More specifically, $W_{D_\phi} = W_2^{L^2}$ when $\phi(x) = \|x\|_2^2$, and $W_{D_\phi} = W_{KL}$ when $\phi(x) = -x^\top \log(x)$.

Probabilistic Properties

In this section, we establish several probabilistic properties of RW divergence. Recall that the Wasserstein distance is controlled by weighted Total Variation (TV) distance (see Theorem 6.15 (Villani 2008) for more details). In parallel, we show that the RW divergence is dominated by the weighted TV distance and the squared Wasserstein- L^2 divergence.

Definition 3.2.2. *The Total Variation distance between the probability distributions \mathbb{P} and \mathbb{Q} is defined as*

$$TV(\mathbb{P}, \mathbb{Q}) := \sup_A |\mathbb{P}(A) - \mathbb{Q}(A)|, \quad (3.1)$$

where A is a Borel set.

Theorem 3.2.1. *Assume that $\phi : \mathcal{X} \rightarrow \mathbb{R}$ is a strictly convex and twice-differentiable function with an L -Lipschitz continuous gradient, then*

$$W_{D_\phi}(\mathbb{P}, \mathbb{Q}) \leq L [\text{diam}(\mathcal{X})]^2 \cdot TV(\mathbb{P}, \mathbb{Q}), \quad (3.2)$$

$$W_{D_\phi}(\mathbb{P}, \mathbb{Q}) \leq \frac{1}{2} L \cdot \left[W_2^{L^2}(\mathbb{P}, \mathbb{Q}) \right]^2, \quad (3.3)$$

where \mathbb{P} and \mathbb{Q} are two probability distributions supported on a compact set $\mathcal{X} \subset \mathbb{R}^d$.

Proof. For the inequality (3.2), define π as the transfer plan that keeps all the mass shared by \mathbb{P} and \mathbb{Q} fixed and distributes the rest uniformly, i.e.,

$$\pi(dx, dy) = (\mathbb{P} \wedge \mathbb{Q})(dx) \delta_{\{y=x\}} + \frac{1}{a} (\mathbb{P} - \mathbb{Q})_+(dx) \cdot (\mathbb{P} - \mathbb{Q})_-(dy),$$

where $\mathbb{P} \wedge \mathbb{Q} = \mathbb{P} - (\mathbb{P} - \mathbb{Q})_+$ and $a = (\mathbb{P} - \mathbb{Q})_+[\mathcal{X}] = (\mathbb{P} - \mathbb{Q})_-[\mathcal{X}]$. Then

$$\begin{aligned}
 W_{D_\phi}(\mathbb{P}, \mathbb{Q}) &\leq \int_{\mathcal{X} \times \mathcal{X}} D_\phi(x, y) \pi(dx, dy) \\
 &= \frac{1}{a} \int_{\mathcal{X} \times \mathcal{X}} [\phi(x) - \phi(y) - \langle \nabla \phi(y), x - y \rangle] (\mathbb{P} - \mathbb{Q})_+(dx) \cdot (\mathbb{P} - \mathbb{Q})_-(dy) \\
 &= \frac{1}{a} \int_{\mathcal{X} \times \mathcal{X}} \left[\int_0^1 \langle \nabla \phi(tx + (1-t)y) - \nabla \phi(y), x - y \rangle dt \right] (\mathbb{P} - \mathbb{Q})_+(dx) \cdot (\mathbb{P} - \mathbb{Q})_-(dy) \\
 &\leq \frac{1}{a} \int_{\mathcal{X} \times \mathcal{X}} \left[\left(\int_0^1 t dt \right) L \|x - y\|_2^2 \right] (\mathbb{P} - \mathbb{Q})_+(dx) (\mathbb{P} - \mathbb{Q})_-(dy) \\
 &\leq \frac{L}{2a} \int_{\mathcal{X} \times \mathcal{X}} [\|x - y\|_2^2] (\mathbb{P} - \mathbb{Q})_+(dx) (\mathbb{P} - \mathbb{Q})_-(dy) \\
 &\leq \frac{L}{a} \int_{\mathcal{X} \times \mathcal{X}} [\|x - x_0\|_2^2 + \|x_0 - y\|_2^2] (\mathbb{P} - \mathbb{Q})_+(dx) (\mathbb{P} - \mathbb{Q})_-(dy) \\
 &\leq L \left[\int_{\mathcal{X}} \|x - x_0\|_2^2 (\mathbb{P} - \mathbb{Q})_+(dx) + \int_{\mathcal{X}} \|y - x_0\|_2^2 (\mathbb{P} - \mathbb{Q})_-(dy) \right] \\
 &= L \int_{\mathcal{X}} \|x - x_0\|_2^2 |\mathbb{P} - \mathbb{Q}|(dx) = L [\text{diam}(\mathcal{X})]^2 \cdot |\mathbb{P}(\mathcal{X}) - \mathbb{Q}(\mathcal{X})| \\
 &\leq L [\text{diam}(\mathcal{X})]^2 \cdot TV(\mathbb{P}, \mathbb{Q}),
 \end{aligned}$$

where the first inequality comes from Definition 3.2.1, the first equality from Definition 2.1.1 and the definition of the specific π , the second inequality is by Lemma 2.1.2, the fourth inequality by the triangle inequality, and the last inequality by Definition 3.2.2.

For the inequality (3.3), we have

$$\begin{aligned}
 W_{D_\phi}(\mathbb{P}, \mathbb{Q}) &= \inf_{\pi \in \Pi(\mathbb{P}, \mathbb{Q})} \int_{\mathcal{X} \times \mathcal{X}} D_\phi(x, y) \pi(dx, dy) \\
 &\leq \frac{1}{2} L \cdot \inf_{\pi \in \Pi(\mathbb{P}, \mathbb{Q})} \int_{\mathcal{X} \times \mathcal{X}} \|x - y\|_2^2 \pi(dx, dy) \\
 &= \frac{1}{2} \cdot \left[W_2^{L^2}(\mathbb{P}, \mathbb{Q}) \right]^2,
 \end{aligned}$$

where the inequality holds thanks to Lemma 2.1.2 and the fact that $\pi(dx, dy) \geq 0$ for any coupling $\pi \in \Pi(\mathbb{P}, \mathbb{Q})$. \square

Next, we establish another key probabilistic property of RW divergence, i.e., the nonasymptotic moment estimates and the concentration inequality. To begin, define two statistics

$$M_q(\mathbb{P}_r) = \int_{\mathcal{X}} \|x\|_2^q \mathbb{P}_r(dx), \quad \text{and} \quad \mathcal{E}_{\alpha, \gamma}(\mathbb{P}_r) = \int_{\mathcal{X}} \exp(\gamma \|x\|_2^\alpha) \mathbb{P}_r(dx).$$

Theorem 3.2.2 (Nonasymptotic Moment Estimate). *Assume that $M_q(\mathbb{P}_r) < +\infty$ for some $q > 2$, then there exists a constant $C(q, d) > 0$ such that, for $n \geq 1$,*

$$\mathbb{E} [W_{D_\phi}(\mathbb{P}_n, \mathbb{P}_r)] \leq \frac{C(q, d) L M_q^{\frac{2}{q}}(\mathbb{P}_r)}{2} \cdot \begin{cases} n^{-\frac{1}{2}} + n^{-\frac{q-2}{q}}, & 1 \leq d \leq 3, q \neq 4, \\ n^{-\frac{1}{2}} \log(1+n) + n^{-\frac{q-2}{q}}, & d = 4, q \neq 4, \\ n^{-\frac{2}{d}} + n^{-\frac{q-2}{q}}, & d \geq 5, q \neq d/(d-2). \end{cases}$$

Theorem 3.2.3 (Concentration Inequality). *Assume one of the following three conditions holds,*

$$\text{Either } \exists \alpha > 2, \exists \gamma > 0, \text{ such that } \mathcal{E}_{\alpha, \gamma}(\mathbb{P}_r) < \infty, \quad (3.4)$$

$$\text{or } \exists \alpha \in (0, 2), \exists \gamma > 0, \text{ such that } \mathcal{E}_{\alpha, \gamma}(\mathbb{P}_r) < \infty, \quad (3.5)$$

$$\text{or } \exists q > 4, \text{ such that } M_q(\mathbb{P}_r) < \infty. \quad (3.6)$$

Then for $n \geq 1$ and $\epsilon > 0$,

$$\mathbb{P}_r (W_{D_\phi}(\mathbb{P}_n, \mathbb{P}_r) \geq \epsilon) \leq a(n, \epsilon) 1_{\{\epsilon \leq \frac{L}{2}\}} + b(n, \epsilon),$$

where

$$a(n, \epsilon) = C_1 \begin{cases} \exp\left(-\frac{4cn\epsilon^2}{L^2}\right), & 1 \leq d \leq 3, \\ \exp\left(-\frac{4cn\epsilon^2}{L^2} \log^2\left(2 + \frac{L}{2\epsilon}\right)\right), & d = 4, \\ \exp\left(-cn\left(\frac{2\epsilon}{L}\right)^{\frac{d}{2}}\right), & d \geq 5, \end{cases}$$

and

$$b(n, \epsilon) = C_2 \begin{cases} \exp\left(-cn\left(\frac{2\epsilon}{L}\right)^{\frac{\alpha}{2}}\right) \cdot 1_{\{\epsilon > \frac{L}{2}\}}, & \text{under condition (3.4),} \\ \exp\left(-c\left(\frac{2n\epsilon}{L}\right)^{\frac{\alpha-\epsilon}{2}}\right) \cdot 1_{\{\epsilon \leq \frac{L}{2}\}} + \exp\left(-c\left(\frac{2n\epsilon}{L}\right)^{\frac{\alpha}{2}}\right) \cdot 1_{\{\epsilon > \frac{L}{2}\}}, & 0 < \epsilon < \alpha, \text{ under condition (3.5),} \\ n\left(\frac{2n\epsilon}{L}\right)^{-\frac{q-\epsilon}{2}}, & 0 < \epsilon < q, \text{ under condition (3.6).} \end{cases}$$

where c, C_1 and C_2 are constants depending on q and d .

Theorem 3.2.2 and Theorem 3.2.3 show that the importance of Lipschitz constant L of the underlying function ϕ in the statistical behaviour of RW divergence. The proof follows from Theorem 1 and Theorem 2 presented in (Fournier and Guillin 2015) and Theorem 3.2.1 in this chapter.

Continuity, Differentiability and Duality Representation

In this section, we establish the continuity, differentiability and duality representation of RW divergence, demonstrating that RW divergence is a reasonable choice for the GANs. We first present a simple yet important lemma.

Lemma 3.2.4 (Decomposition of RW divergence). *The RW divergence can be decomposed in terms of the distorted squared Wasserstein- L_2 divergence of order 2 with several additional residual terms independent of the choice of coupling π , i.e.,*

$$\begin{aligned} W_{D_\phi}(\mathbb{P}, \mathbb{Q}) &= \frac{1}{2} \left[W_2^{L^2}(\mathbb{P}, \mathbb{Q} \circ (\nabla\phi)^{-1}) \right]^2 \\ &\quad + \int_{\mathcal{X}} \left[\phi(x) - \frac{1}{2} \|x\|_2^2 \right] \mathbb{P}(dx) + \int_{\mathcal{X}} \left[\langle \nabla\phi(x), x \rangle - \phi(x) - \frac{1}{2} \|\nabla\phi(x)\|_2^2 \right] \mathbb{Q}(dx). \end{aligned}$$

See Figure 3.1.

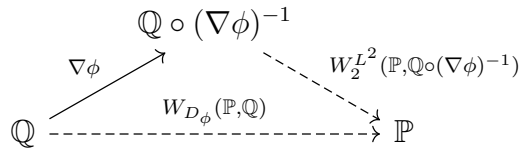


Figure 3.1: The decomposition of W_{D_ϕ} where the solid arrow denotes transformation and the dashed arrows denote the divergences between probability distributions.

Proof. First, we need to prove that the inverse of ϕ is well-defined. Since $\nabla^2\phi(x) \succ 0, \forall x \in \mathcal{X}$, the gradient mapping $\nabla\phi : \mathcal{X} \rightarrow \mathbb{R}^d$ has a positive-definite Jacobian matrix at each point. Applying the mean value theorem yields that ϕ is injective so the inverse of $\nabla\phi$ exists and is bijective. Denote it as

$$(\nabla\phi)^{-1} : \nabla\phi(\mathcal{X}) \rightarrow \mathcal{X},$$

then

$$\mathbb{Q} \circ (\nabla\phi)^{-1} : \mathbb{R}^d \rightarrow \mathbb{R}$$

is also a probability distribution. Thus

$$\begin{aligned} W_{D_\phi}(\mathbb{P}, \mathbb{Q}) &= \inf_{\pi \in \Pi(\mathbb{P}, \mathbb{Q})} \int_{\mathcal{X} \times \mathcal{X}} [\phi(x) - \phi(y) - \langle \nabla\phi(y), x - y \rangle] \pi(dx, dy) \\ &= \inf_{\pi \in \Pi(\mathbb{P}, \mathbb{Q})} \int_{\mathcal{X} \times \mathcal{X}} \left[\frac{1}{2} \|x\|_2^2 + \frac{1}{2} \|\nabla\phi(y)\|_2^2 - \langle \nabla\phi(y), x \rangle \right] \pi(dx, dy) \\ &\quad + \int_{\mathcal{X} \times \mathcal{X}} \left[\phi(x) - \frac{1}{2} \|x\|_2^2 \right] \pi(dx, dy) \\ &\quad + \int_{\mathcal{X} \times \mathcal{X}} \left[\langle \nabla\phi(y), y \rangle - \phi(y) - \frac{1}{2} \|\nabla\phi(y)\|_2^2 \right] \pi(dx, dy) \\ &= \inf_{\pi \in \Pi(\mathbb{P}, \mathbb{Q})} \int_{\mathcal{X} \times \mathcal{X}} \left[\frac{1}{2} \|x\|_2^2 + \frac{1}{2} \|\nabla\phi(y)\|_2^2 - \langle \nabla\phi(y), x \rangle \right] \pi(dx, dy) \\ &\quad + \int_{\mathcal{X}} \left[\phi(x) - \frac{1}{2} \|x\|_2^2 \right] \mathbb{P}(dx) + \int_{\mathcal{X}} \left[\langle \nabla\phi(x), x \rangle - \phi(x) - \frac{1}{2} \|\nabla\phi(x)\|_2^2 \right] \mathbb{Q}(dx). \end{aligned}$$

Furthermore,

$$\begin{aligned}
 \left[W_2^{L^2}(\mathbb{P}, \mathbb{Q} \circ (\nabla\phi)^{-1}) \right]^2 &= \inf_{\pi \in \Pi(\mathbb{P}, \mathbb{Q} \circ (\nabla\phi)^{-1})} \int_{\mathcal{X} \times \mathbb{R}^d} \|x - y\|_2^2 \pi(dx, dy) \\
 &= \inf_{\pi \in \Pi(\mathbb{P}, \mathbb{Q})} \int_{\mathcal{X} \times \mathbb{R}^d} \|x - \nabla\phi(y)\|_2^2 \pi(dx, dy) \\
 &= \inf_{\pi \in \Pi(\mathbb{P}, \mathbb{Q})} \int_{\mathcal{X} \times \mathcal{X}} [\|x\|_2^2 + \|\nabla\phi(y)\|_2^2 - 2\langle \nabla\phi(y), x \rangle] \pi(dx, dy).
 \end{aligned}$$

Therefore,

$$\begin{aligned}
 W_{D_\phi}(\mathbb{P}, \mathbb{Q}) &= \frac{1}{2} \left[W_2^{L^2}(\mathbb{P}, \mathbb{Q} \circ (\nabla\phi)^{-1}) \right]^2 \\
 &\quad + \int_{\mathcal{X}} \left[\phi(x) - \frac{1}{2} \|x\|_2^2 \right] \mathbb{P}(dx) + \int_{\mathcal{X}} \left[\langle \nabla\phi(x), x \rangle - \phi(x) - \frac{1}{2} \|\nabla\phi(x)\|_2^2 \right] \mathbb{Q}(dx).
 \end{aligned}$$

□

Now we are ready to present our main results on the continuity and differentiability of the parametrized RW divergence in the generative modeling.

Definition 3.2.3 (Generative modeling). *The procedure of generative modeling is to approximate an unknown probability distribution \mathbb{P}_r by constructing a class of suitable parametric probability distributions \mathbb{P}_θ . More specifically, define a latent variable $Z \in \mathcal{Z}$ with a fixed probability distribution \mathbb{P}_Z and a sequence of parametric functions $g_\theta : \mathcal{Z} \rightarrow \mathcal{X}$. Then \mathbb{P}_θ is defined as the probability distribution of $g_\theta(Z)$.*

Theorem 3.2.5 (Continuity and Differentiability of RW divergence). 1. $W_{D_\phi}(\mathbb{P}_r, \mathbb{P}_\theta)$ is continuous in θ if g_θ is continuous in θ .

2. $W_{D_\phi}(\mathbb{P}_r, \mathbb{P}_\theta)$ is differentiable almost everywhere if g_θ is locally Lipschitz with a constant $L(\theta, z)$ such that $\mathbb{E}[L(\theta, Z)^2] < \infty$, i.e., for each given (θ_0, z_0) , there exists a neighborhood \mathcal{N} such that

$$\|g_\theta(z) - g_{\theta_0}(z_0)\|_2 \leq L(\theta_0, z_0) (\|\theta - \theta_0\|_2 + \|z - z_0\|_2).$$

for any $(\theta, z) \in \mathcal{N}$.

Proof. It follows from Lemma 3.2.4 that $W_{D_\phi}(\mathbb{P}_r, \mathbb{P}_\theta) = T_1 + T_2$, where

$$\begin{aligned}
 T_1 &= \frac{1}{2} \left[W_2^{L^2}(\mathbb{P}_r, \mathbb{P}_\theta \circ (\nabla\phi)^{-1}) \right]^2, \\
 T_2 &= \int_{\mathcal{X}} \left[\phi(x) - \frac{1}{2} \|x\|_2^2 \right] \mathbb{P}_r(dx) + \int_{\mathcal{X}} \left[\langle \nabla\phi(x), x \rangle - \phi(x) - \frac{1}{2} \|\nabla\phi(x)\|_2^2 \right] \mathbb{P}_\theta(dx).
 \end{aligned}$$

We observe that T_2 is continuous and differentiable with respect to θ since ϕ is a twice differentiable function. Furthermore, since $(\nabla\phi)^{-1}$ is also continuous and differentiable, it

suffices to show that $W_2^{L^2}(\mathbb{P}_r, \mathbb{P}_\theta)$ is continuous in θ if g_θ is continuous in θ , and differentiable almost everywhere if g_θ is locally Lipschitz with a constant $L(\theta, z)$ such that $\mathbb{E}[L(\theta, Z)^2] < \infty$ for any θ .

Given two vectors $\theta_0, \theta \in \mathbb{R}^d$, we define π as a joint distribution of $(g_\theta(Z), g_{\theta_0}(Z))$ where $Z \sim \mathbb{P}_Z$, then

$$\begin{aligned} W_2^{L^2}(\mathbb{P}_\theta, \mathbb{P}_{\theta_0}) &\leq \left(\int_{\mathcal{X} \times \mathcal{X}} \|x - y\|_2^2 \pi(dx, dy) \right)^{1/2} \\ &= \left(\int_{\mathcal{Z}} \|g_\theta(z) - g_{\theta_0}(z)\|_2^2 \mathbb{P}_Z(dz) \right)^{1/2}, \end{aligned}$$

where $\|g_\theta(z) - g_{\theta_0}(z)\|_2^2 \rightarrow 0, \forall z \in \mathcal{Z}$, since g_θ is continuous in θ . Furthermore, $\|g_{\theta_1}(z) - g_{\theta_2}(z)\|_2^2$ is uniformly bounded on \mathcal{Z} since $g_\theta(x) \in \mathcal{X}$ and \mathcal{X} is a compact set. Therefore, applying the bounded convergence theorem yields

$$\begin{aligned} \left| W_2^{L^2}(\mathbb{P}_r, \mathbb{P}_\theta) - W_2^{L^2}(\mathbb{P}_r, \mathbb{P}_{\theta_0}) \right| &\leq W_2^{L^2}(\mathbb{P}_\theta, \mathbb{P}_{\theta_0}) \\ &\leq \left(\int_{\mathcal{Z}} \|g_\theta(z) - g_{\theta_0}(z)\|_2^2 \mathbb{P}_Z(dz) \right)^{1/2} \rightarrow 0, \quad \text{as } \theta \rightarrow \theta_0. \end{aligned}$$

where the first inequality comes from the triangle inequality.

Given a pair (θ_0, z_0) , the local Lipschitz continuity of g_θ implies that there exists a neighborhood \mathcal{N} such that $\|g_\theta(z) - g_{\theta_0}(z_0)\|_2 \leq L(\theta_0, z_0) (\|\theta - \theta_0\|_2 + \|z - z_0\|_2)$ for any $(\theta, z) \in \mathcal{N}$. Then

$$\begin{aligned} \int_{\mathcal{Z}} \|g_\theta(z_0) - g_{\theta_0}(z_0)\|_2^2 \mathbb{P}_Z(dz_0) &\leq \int_{\mathcal{Z}} [L(\theta_0, z_0)]^2 \cdot \|\theta - \theta_0\|_2^2 \mathbb{P}_Z(dz_0) \\ &= \|\theta - \theta_0\|_2^2 \cdot \mathbb{E}[L(\theta_0, Z)^2]. \end{aligned}$$

Therefore,

$$\begin{aligned} \left| W_2^{L^2}(\mathbb{P}_r, \mathbb{P}_\theta) - W_2^{L^2}(\mathbb{P}_r, \mathbb{P}_{\theta_0}) \right| &\leq W_2^{L^2}(\mathbb{P}_\theta, \mathbb{P}_{\theta_0}) \\ &\leq \left(\int_{\mathcal{Z}} \|g_\theta(z_0) - g_{\theta_0}(z_0)\|_2^2 \mathbb{P}_Z(dz_0) \right)^{1/2} \\ &\leq \|\theta - \theta_0\|_2 \cdot \mathbb{E}[L(\theta, Z)^2]^{1/2}, \end{aligned}$$

which implies that $W_2^{L^2}(\mathbb{P}_r, \mathbb{P}_\theta)$ is locally Lipschitz. Applying the Rademacher's theorem (Evans and Gariepy 2015) yields that $W_2^{L^2}(\mathbb{P}_r, \mathbb{P}_\theta)$ is differentiable with respect to θ almost everywhere. \square

Next is the the duality representation of RW divergence.

Theorem 3.2.6 (Duality Representation of RW divergence). *Given two probability distributions \mathbb{P} and \mathbb{Q} such that*

$$\int_{\mathcal{X}} \|x\|_2^2 (\mathbb{P} + \mathbb{Q})(dx) < +\infty,$$

then there exists a Lipschitz continuous function $f : \mathcal{X} \rightarrow \mathbb{R}$ such that the RW divergence has the following duality representation

$$W_{D_\phi}(\mathbb{P}, \mathbb{Q}) = \int_{\mathcal{X}} \phi(x) (\mathbb{P} - \mathbb{Q})(dx) + \int_{\mathcal{X}} \langle \nabla \phi(x), x \rangle \mathbb{Q}(dx) - \left(\int_{\mathcal{X}} f(x) \mathbb{P}(dx) + \int_{\mathcal{X}} f^*(\nabla \phi(x)) \mathbb{Q}(dx) \right),$$

where f^* is the conjugate of f , such that $f^*(y) = \sup_{x \in \mathbb{R}^d} \langle x, y \rangle - f(x)$.

Proof. First,

$$\left[W_2^{L^2}(\mathbb{P}, \mathbb{Q}) \right]^2 = \inf_{\pi \in \Pi(\mathbb{P}, \mathbb{Q})} \int_{\mathcal{X} \times \mathcal{X}} \|x - y\|_2^2 \pi(dx, dy) \quad (3.7)$$

$$= \int_{\mathcal{X}} \|x\|_2^2 (\mathbb{P} + \mathbb{Q})(dx) - \sup_{\pi \in \Pi(\mathbb{P}, \mathbb{Q})} \int_{\mathcal{X} \times \mathcal{X}} 2x^\top y \pi(dx, dy), \quad (3.8)$$

then it follows from Proposition 3.1 (Brenier 1991) that there exists a Lipschitz continuous function $f : \mathcal{X} \rightarrow \mathbb{R}$ such that the squared Wasserstein- L_2 divergence of order 2 has a duality representation:

$$\begin{aligned} \left[W_2^{L^2}(\mathbb{P}, \mathbb{Q}) \right]^2 &= \inf_{\pi \in \Pi(\mathbb{P}, \mathbb{Q})} \int_{\mathcal{X} \times \mathcal{X}} \|x - y\|_2^2 \pi(dx, dy) \\ &= \int_{\mathcal{X}} \|x\|_2^2 (\mathbb{P} + \mathbb{Q})(dx) - 2 \left(\int_{\mathcal{X}} f(x) \mathbb{P}(dx) + \int_{\mathcal{X}} f^*(x) \mathbb{Q}(dx) \right), \end{aligned}$$

where $f^*(y) = \sup_{x \in \mathbb{R}^d} \langle x, y \rangle - f(x)$. By Lemma 3.2.4,

$$\begin{aligned} W_{D_\phi}(\mathbb{P}, \mathbb{Q}) &= \frac{1}{2} \left[W_2^{L^2}(\mathbb{P}, \mathbb{Q} \circ (\nabla \phi)^{-1}) \right]^2 \\ &\quad + \int_{\mathcal{X}} \left[\phi(x) - \frac{1}{2} \|x\|_2^2 \right] \mathbb{P}(dx) + \int_{\mathcal{X}} \left[\langle \nabla \phi(x), x \rangle - \phi(x) - \frac{1}{2} \|\nabla \phi(x)\|_2^2 \right] \mathbb{Q}(dx), \\ &= \frac{1}{2} \left(\int_{\mathcal{X}} \|x\|_2^2 \mathbb{P}(dx) + \int_{\mathcal{X}} \|\nabla \phi(x)\|_2^2 \mathbb{Q}(dx) \right) \\ &\quad - \left(\int_{\mathcal{X}} f(x) \mathbb{P}(dx) + \int_{\mathcal{X}} f^*(\nabla \phi(x)) \mathbb{Q}(dx) \right) \\ &\quad + \int_{\mathcal{X}} \left[\phi(x) - \frac{1}{2} \|x\|_2^2 \right] \mathbb{P}(dx) + \int_{\mathcal{X}} \left[\langle \nabla \phi(x), x \rangle - \phi(x) - \frac{1}{2} \|\nabla \phi(x)\|_2^2 \right] \mathbb{Q}(dx) \\ &= \int_{\mathcal{X}} \phi(x) (\mathbb{P} - \mathbb{Q})(dx) + \int_{\mathcal{X}} \langle \nabla \phi(x), x \rangle \mathbb{Q}(dx) \\ &\quad - \left(\int_{\mathcal{X}} f(x) \mathbb{P}(dx) + \int_{\mathcal{X}} f^*(\nabla \phi(x)) \mathbb{Q}(dx) \right). \end{aligned}$$

□

Finally, we show that Theorem 3.2.6 allows for an explicit formula for the gradient evaluation in the generative modeling (Definition 3.2.3), providing the theoretical guarantee for the RWGANs training.

Corollary 3.2.6.1 (Gradient Evaluation). *Under the setting of generative modeling, we assume that g_θ is locally Lipschitz with a constant $L(\theta, z)$ such that $\mathbb{E}[L(\theta, Z)^2] < \infty$ and*

$$\int_{\mathcal{X}} \|x\|_2^2 (\mathbb{P}_r + \mathbb{P}_\theta)(dx) < +\infty.$$

Then there exists a Lipschitz continuous solution $f : \mathcal{X} \rightarrow \mathbb{R}$ such that the gradient of the RW divergence has an explicit form of

$$\nabla_\theta [W_{D_\phi}(\mathbb{P}_r, \mathbb{P}_\theta)] = \mathbb{E}_Z \left[[\nabla_\theta g_\theta(Z)]^\top \nabla^2 \phi(g_\theta(Z)) g_\theta(Z) \right] + \mathbb{E}_Z [\nabla_\theta f(\nabla \phi(g_\theta(Z)))].$$

Proof. Since g_θ is a locally Lipschitz and $\int_{\mathcal{X}} \|x\|_2^2 (\mathbb{P}_r + \mathbb{P}_\theta)(dx) < +\infty$, it follows from Theorem 3.2.5 and Theorem 3.2.6 that $W_{D_\phi}(\mathbb{P}_r, \mathbb{P}_\theta)$ is differentiable almost everywhere and there exists a Lipschitz continuous function $\tilde{f} : \mathcal{X} \rightarrow \mathbb{R}$ such that the RW divergence has a duality representation as

$$\begin{aligned} W_{D_\phi}(\mathbb{P}_r, \mathbb{P}_\theta) &= \int_{\mathcal{X}} \phi(x) (\mathbb{P}_r - \mathbb{P}_\theta)(dx) + \int_{\mathcal{X}} \langle \nabla \phi(x), x \rangle \mathbb{P}_\theta(dx) \\ &\quad - \left(\int_{\mathcal{X}} \tilde{f}(x) \mathbb{P}_r(dx) + \int_{\mathcal{X}} \tilde{f}^*(\nabla \phi(x)) \mathbb{P}_\theta(dx) \right). \end{aligned}$$

By the envelope theorem (Milgrom and Segal 2002), we obtain that

$$\begin{aligned} \nabla_\theta [W_{D_\phi}(\mathbb{P}_r, \mathbb{P}_\theta)] &= \nabla_\theta \left[- \int_{\mathcal{X}} \phi(x) \mathbb{P}_\theta(dx) + \int_{\mathcal{X}} \langle \nabla \phi(x), x \rangle \mathbb{P}_\theta(dx) - \int_{\mathcal{X}} \tilde{f}^*(\nabla \phi(x)) \mathbb{P}_\theta(dx) \right] \\ &= \nabla_\theta \left[- \int_{\mathcal{Z}} \phi(g_\theta(z)) \mathbb{P}_Z(dz) + \int_{\mathcal{Z}} \langle \nabla \phi(g_\theta(z)), g_\theta(z) \rangle \mathbb{P}_Z(dz) \right. \\ &\quad \left. - \int_{\mathcal{Z}} \tilde{f}^*(\nabla \phi(g_\theta(z))) \mathbb{P}_Z(dz) \right] \\ &= - \int_{\mathcal{Z}} [\nabla_\theta g_\theta(z)]^\top \nabla \phi(g_\theta(z)) \mathbb{P}_Z(dz) + \int_{\mathcal{Z}} [\nabla_\theta g_\theta(z)]^\top \nabla \phi(g_\theta(z)) \mathbb{P}_Z(dz) \\ &\quad + \int_{\mathcal{Z}} [\nabla_\theta g_\theta(z)]^\top \nabla^2 \phi(g_\theta(z)) g_\theta(z) \mathbb{P}_Z(dz) - \int_{\mathcal{Z}} \nabla_\theta \tilde{f}^*(\nabla \phi(g_\theta(z))) \mathbb{P}_Z(dz) \\ &= \int_{\mathcal{Z}} [\nabla_\theta g_\theta(z)]^\top \nabla^2 \phi(g_\theta(z)) g_\theta(z) \mathbb{P}_Z(dz) - \int_{\mathcal{Z}} \nabla_\theta \tilde{f}^*(\nabla \phi(g_\theta(z))) \mathbb{P}_Z(dz) \end{aligned}$$

Letting $f = -\tilde{f}^*$,

$$\nabla_\theta [W_{D_\phi}(\mathbb{P}_r, \mathbb{P}_\theta)] = \mathbb{E}_Z \left[[\nabla_\theta g_\theta(Z)]^\top \nabla^2 \phi(g_\theta(Z)) g_\theta(Z) \right] + \mathbb{E}_Z [\nabla_\theta f(\nabla \phi(g_\theta(Z)))],$$

where f is Lipschitz continuous. □

3.3 Experiments

RWGANs

In this section, we will present numerical evaluation on image generations to demonstrate the effectiveness and efficiency of using RW divergence in GANs. For the review of the basics of GANs, please refer to Section 2.4). For the review of structures of WGAN and WGAN-GP, please refer to Section 2.4. We first derive the computation of the gradient of RW in training GANs (RWGANs) (Section 3.3). We then describe our experiment framework and settings (Section 3.3). Finally we report the experimental results under RWGANs versus other well-established variants of GANs including WGAN, and WGAN-GP in Section 3.3.

Gradient descent and smoothness of RW divergence

In the training of GANs, descent methods are typically used to minimize Equation (2.6). Similarly, differentiability is needed for RW divergence in the RWGANs approach. As in WGANs, despite the theoretical explicit formulas derived in the duality representation and the gradient evaluation (Theorem 3.2.6 and Corollary 3.2.6.1), it is infeasible to directly compute such an f in practice. Nevertheless, since the RW divergence is parametrized by any strictly convex function in RWGANs, we obtain a great deal of flexibility in the choice of loss functions. For example, one can choose an appropriate ϕ such that

$$\nabla_{\theta} [W_{D_{\phi}}(\mathbb{P}_r, \mathbb{P}_{\theta})] \approx \mathbb{E}_Z [\nabla_{\theta} f(\nabla \phi(g_{\theta}(z)))].$$

For instance, one can try the KL divergence where $\nabla^2 \phi(x) = \text{diag}(1/x)$, observing that

$$\mathbb{E}_Z \left[[\nabla_{\theta} g_{\theta}(Z)]^{\top} \nabla^2 \phi(g_{\theta}(Z)) g_{\theta}(Z) \right] = \mathbb{E}_Z \left[[\nabla_{\theta} g_{\theta}(Z)]^{\top} \bar{\mathbf{I}} \right] \leq C,$$

where C is a constant depending on the Lipschitz constant of g_{θ} . This implies that this term is controlled by θ during the process of training. The numerical results in Section 3.3 confirm the effectiveness of our heuristic.

Experimental framework and settings

Experimental framework. The similarity between our experimental framework and the one in WGANs (Arjovsky, Chintala, and Bottou 2017) is: we apply back-propagation to train the generator and discriminator networks, and update the parameters once in the generative model and n_{critic} times in the discriminator network.

The differences between ours and the WGANs (Algorithm 1 in Chapter 2) are: 1) we use $\nabla \phi$ to do the asymmetric clipping instead of the symmetric clipping. Note that the asymmetric clipping guarantees the Lipschitz continuity of f and $\nabla \phi(w) \in [-c, c]$; 2) we use a scaling parameter S to stabilize the asymmetric clipping. This parameter ensures that the norm of the gradient is controlled within a small range. This is critical for the experiment

since it reduces the variance of the gradient updates; 3) we adopt RMSProp (Tieleman and Hinton 2012) instead of ADAM (Kingma and Ba 2014), which allows a choice of a larger step-size and avoids the non-stationary problem (Mnih et al. 2016).

The details are described in Algorithm 3, where the boxed equation highlights the asymmetric clipping procedure, one of the key algorithmic differences between WGANs and RWGANs.

Algorithm 3 RWGANs. The default values $\alpha = 0.0005$, $c = 0.005$, $S = 0.01$, $m = 64$, $n_{critic} = 5$.

Require: α : the learning rate; c : the clipping parameter; m : the batch size; n_{critic} , the number of iterations of the critic per generator iteration; N_{max} , the maximum number of one forward pass and one backward pass of all the training examples.

Require: w_0 , initial critic parameters; θ_0 : initial generator’s parameters.

for $N = 1, 2, \dots, N_{max}$ **do**

for $t = 0, \dots, n_{critic}$ **do**

 Sample a batch of real data $\{x_i\}_{i=1}^m$ from \mathbb{P}_r .

 Sample a batch of prior samples $\{z_i\}_{i=1}^m$ from $p(z)$.

$g_w \leftarrow \frac{1}{m} \sum_{i=1}^m [\nabla_w f_w(x_i) - \nabla_w f_w(g_\theta(z_i))]$.

$w \leftarrow w + \alpha \cdot \text{RMSProp}(w, g_w)$.

$w \leftarrow \text{clip}(w, -S \cdot (\nabla\phi)^{-1}(-c), S \cdot (\nabla\phi)^{-1}(c))$.

end for

 Sample a batch of prior samples $\{z_i\}_{i=1}^m$ from $p(z)$.

$g_\theta \leftarrow -\frac{1}{m} \sum_{i=1}^m \nabla_\theta f_w(\nabla\phi(g_\theta(z_i)))$.

$\theta \leftarrow \theta - \alpha \cdot \text{RMSProp}(\theta, g_\theta)$.

end for

Experimental settings. In order to test RWGANs, we adopt nine baseline methods as discussed in the introduction. They are RWGANs, WGANs (Arjovsky, Chintala, and Bottou 2017), WGANs-GP (Gulrajani et al. 2017), CGANs (Mirza and Osindero 2014), InfoGANs (X. Chen et al. 2016), GANs (Goodfellow et al. 2014), LSGANs (Mao et al. 2017), DRAGANs (Kodali et al. 2017), BEGANs (Berthelot, Schumm, and Metz 2017), EBGANs (Zhao, Mathieu, and LeCun 2016), and ACGANs (Odena, Olah, and Shlens 2017). The implementation of all these approaches is based on publicly available online information. In addition, we use the following four standard and well-known datasets in our experiment.

1. MNIST is a dataset of handwritten digits. It has a training set of 60,000 examples, and a test set of 10,000 examples. It is a subset of a larger set available from NIST. The digits have been size-normalized and centered in a fixed-size image.

2. Fashion-MNIST is an alternative dataset of Zalando’s article images to MNIST. It consists of a training set of 60,000 examples and a test set of 10,000 examples. Each example is a 28×28 gray-scale image, associated with a label from 10 classes.
3. The CIFAR-10¹ dataset consists of 60000 32×32 color images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images.
4. The ImageNet² dataset is a large visual database designed for visual object recognition research. As of 2016, over ten million URLs of images have been hand-annotated by ImageNet to indicate which objects are in the picture. In at least one million of the images, bounding boxes are also provided.

Metric. The negative critic loss, well-known as the standard quantitative metric, is used in all our experiments. In addition to the negative critic loss, we use the inception score (Salimans et al. 2016) to evaluate samples generated by three WGANs methods on CIFAR-10 and ImageNet. The inception score is defined as follows:

$$\text{Inception_Score} = \exp \{ \mathbb{E}_x [D_{\text{KL}}(p(y|x), p(y))] \},$$

where $p(y|x)$ is given by the inception network. A high inception score is an indicator that the images generated by the model are highly interpretable and diversified. It is also highly correlated with human evaluation of the images.

Experimental Results.

Experiments on MNIST and Fashion-MNIST: We start our experiment by training models using the ten different GANs procedures on MNIST and Fashion-MNIST. The architecture is DCGAN (Radford, Metz, and Chintala 2015) and the maximum number of epochs is 100.

Figure 3.5 shows the training curves of the negative critic loss of all candidate approaches. The figure indicates that RWGANs and WGANs are stable with the smallest variances, where RWGANs has a slight higher variance partly due to the use of a larger step-size and asymmetric clipping. This slightly higher variance, nevertheless, speeds up the rate of training. Indeed, as illustrated in Figure 3.3 and Figure 3.4, RWGANs is the fastest to generate meaningful images. Note that CGANs and InfoGANs seem faster in generating clearer images, they fall into local optima in the optimization procedure therefore the samples all look similar and not as diverse.

Experiments on CIFAR-10 and ImageNet: After observing that WGANs and RWGANs perform the best among all the variants of GANs, we proceed to compare RWGANs and WGANs, together with WGANs with Gradient Penalty (WGANs-GP), on two much larger datasets CIFAR-10 and ImageNet. Here the architectures used are DCGAN and ReLU-MLP (Conan-Guez and Rossi 2002) and the maximum number of epochs is set to 25.

¹<https://www.cs.toronto.edu/~kriz/cifar.html>

²http://image-net.org/small/train_64x64.tar

Figure 3.10 shows the training curves of the negative critic loss of all candidate approaches again. Except for the small variance of WGANs, we observe that, in terms of the negative loss, WGANs-GP tend to diverge as the training progresses, implying that such method might not be robust in practice despite its fast rate of training. In this case, RWGANs achieve relatively low variance with convergent negative critic loss, leading to a trade-off between robustness and efficiency.

We then evaluate the candidate methods with the inception score and present the results in Table 3.2. The table shows that RWGANs are often the fastest method. They perform the best in three out of four cases during several early epochs, and obtain images with competitively high quality at the final stage. Figures 3.6, Figure 3.7, Figure 3.8 and Figure 3.9 show the sample qualities of the image generated at the initial and final stages, which strongly supports our conclusion.

Architecture	Method	CIFAR-10		ImageNet	
		First 5 epochs	Last 10 epochs	First 3 epochs	Last 5 epochs
DCGAN	RWGANs	1.8606	2.3962	2.0430	2.7008
	WGANs	1.6329	2.4246	2.2070	2.7972
	WGANs-GP	1.7259	2.3731	2.2749	2.7331
MLP	RWGANs	1.3126	2.1710	2.0025	2.4805
	WGANs	1.2798	1.9007	1.7401	2.2304
	WGANs-GP	1.2711	2.2192	1.8845	2.3448

Figure 3.2: Inception scores at the beginning and final stages of training. DCGAN refers to the standard DCGAN generator and MLP refers to an ReLU-MLP with 4 hidden layers and 512 units at each layer.

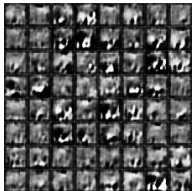



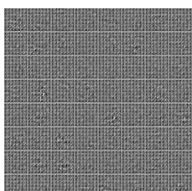
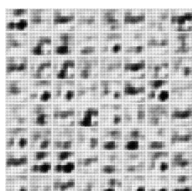






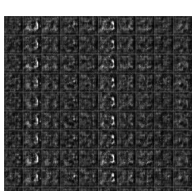
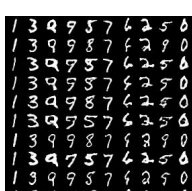


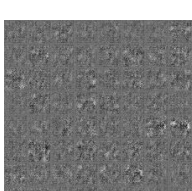
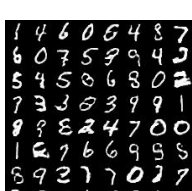

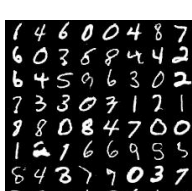
3.4 Discussion

In this chapter, we propose a novel class of statistical divergence called RW divergence and establish several important theoretical properties. Numerical experiments, with RW parametrized by the KL divergence in image generation, show that RWGANs is a promising trade-off between WGANs and WGANs-GP, achieving both the robustness and efficiency during the learning process. The asymmetric clipping in RWGANs is a viable alternative to the gradient penalty and the symmetric clipping in WGANs, avoiding the low-quality samples and the failure of convergence. We also discuss a potential application of RW divergence in the context of robust optimization and explain how it can be used to construct ambiguity sets.

The flexible framework of RW divergences raises a natural question on whether one can select ϕ according to the data and the structure of the problem. This question is partially addressed by Proposition 1 and Theorem 4.1.3 in Chapter 4: with the objective of variance

stabilization, a reasonable choice of the Bregman divergence is the Mahalanobis distance with the corresponding covariance matrix being the estimated Fisher information matrix.

While we highlight only the applications of RW to GANs (and robust optimization in Chapter 5), we believe that the theoretical results of RW divergence can be a valuable addition to the rich theory for optimal transport, where regularities of Wasserstein-based cost functions have been extensively studied (Caffarelli 1991; Caffarelli 1992; S. Chen and Figalli 2017; Villani 2008). With the extension of Bregman divergence to the functional space (Frigyik, Srivastava, and Gupta 2008), the application of RW divergence in martingale optimal transport is also promising.

Method	$N = 1$	$N = 10$	$N = 25$	$N = 100$
RWGANs				
WGANs				
CGANs				
InfoGANs				
GANs				

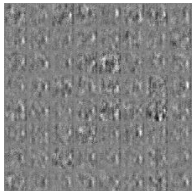



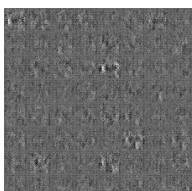



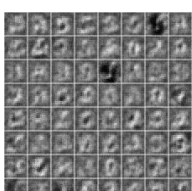



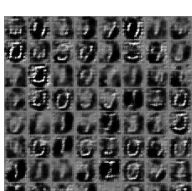
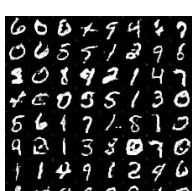


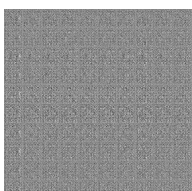
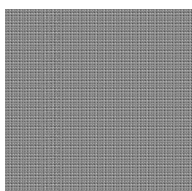
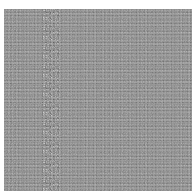

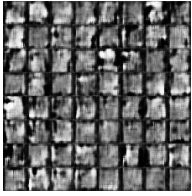



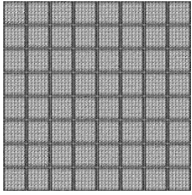
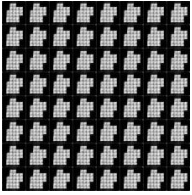




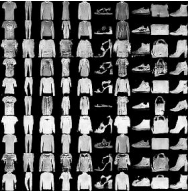
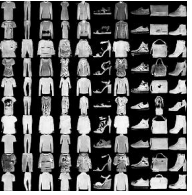
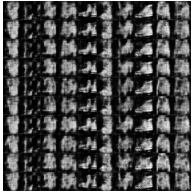
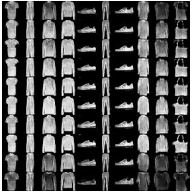
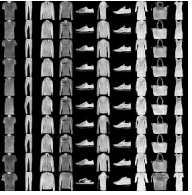
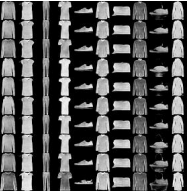
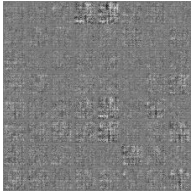
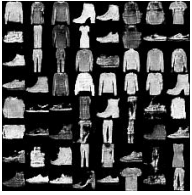


Method	$N = 1$	$N = 10$	$N = 25$	$N = 100$
LSGANs				
DRAGANs				
BEGANs				
EBGANs				
ACGANs				

Figure 3.3: Sample qualities at different stages of training on MNIST.

Method	$N = 1$	$N = 10$	$N = 25$	$N = 100$
RWGANs				
WGANs				
CGANs				
InfoGANs				
GANs				

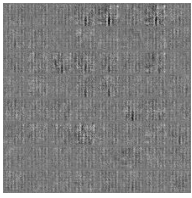



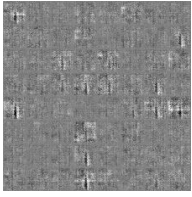



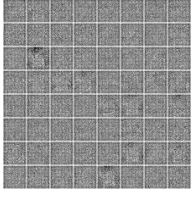
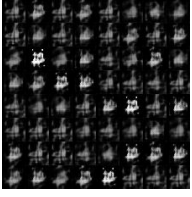

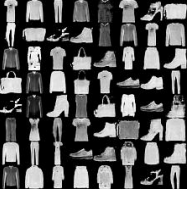
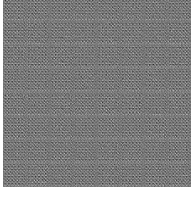
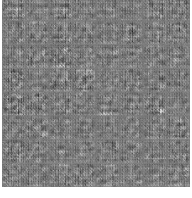


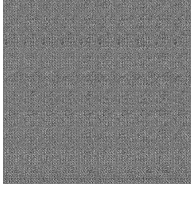
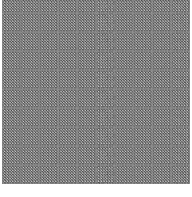
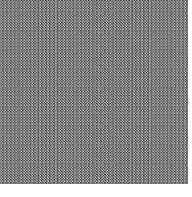
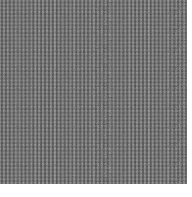
Method	$N = 1$	$N = 10$	$N = 25$	$N = 100$
LSGANs				
DRAGANs				
BEGANs				
EBGANs				
ACGANs				

Figure 3.4: Sample qualities at different stages of training on Fashion-MNIST.

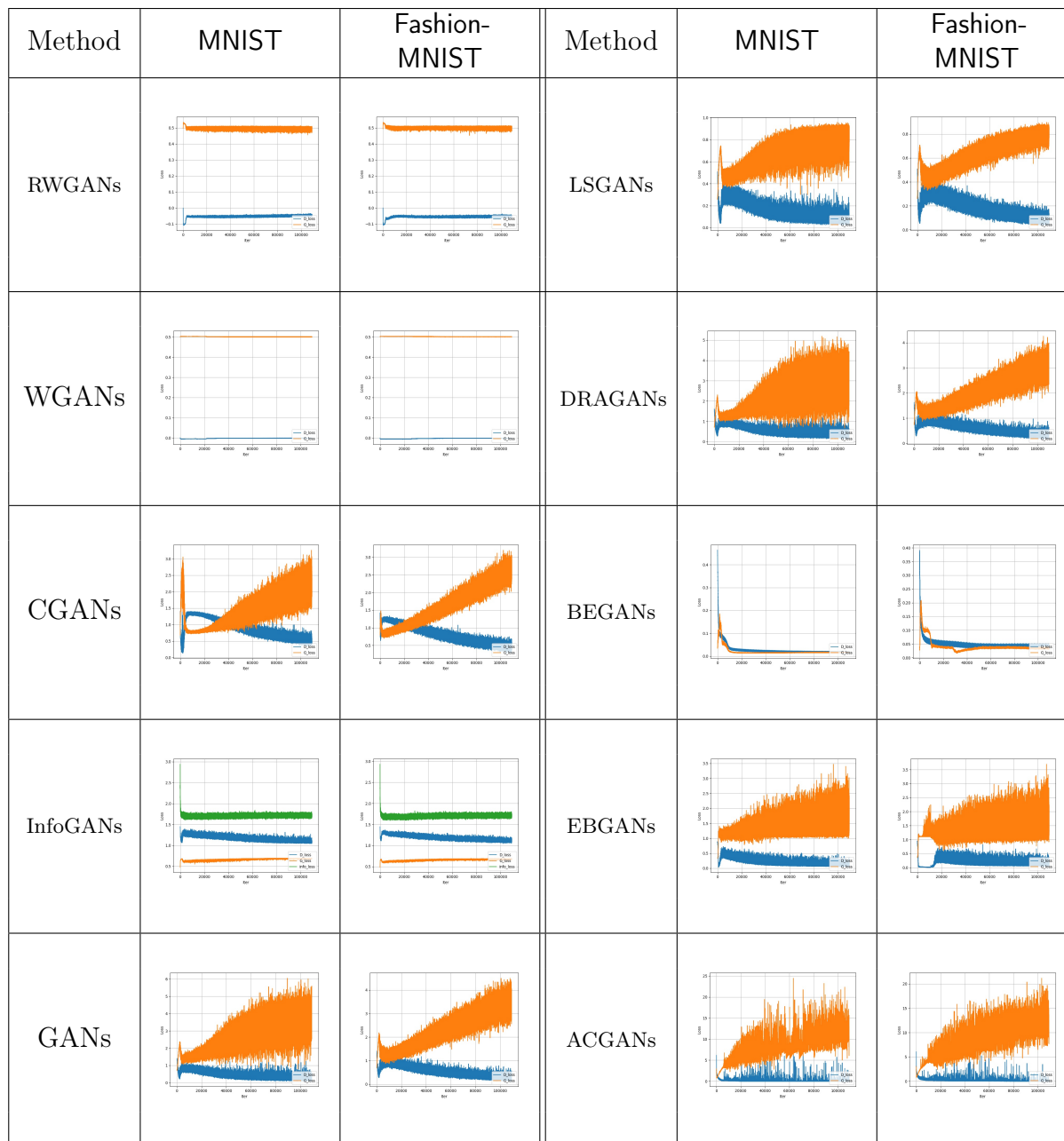


Figure 3.5: Training curves of the negative critic loss at different stages of training on MNIST and Fashion-MNIST. G_{loss} and D_{loss} refer to the loss in generative and discriminative nets, which is plotted in orange and blue lines, respectively.



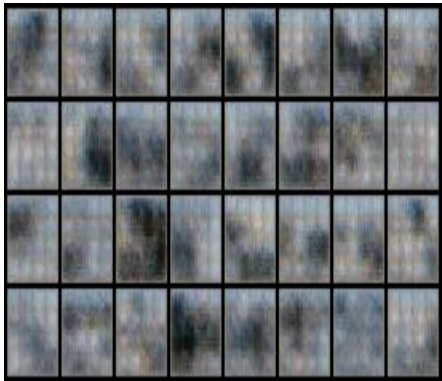
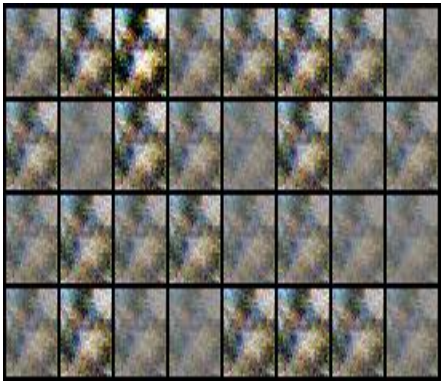

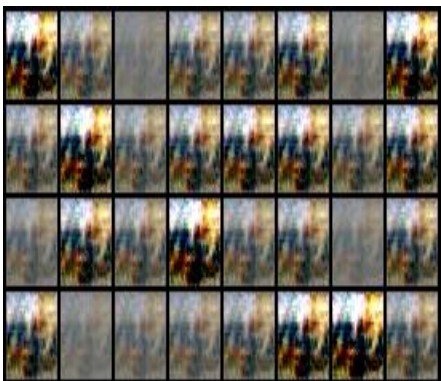
Method	$N = 1$	
	DCGAN	MLP
RWGANs		
WGANs		
WGANs-GP		

Figure 3.6: Sample qualities at the initial stage of training on CIFAR-10.

Method	$N = 100$	
	DCGAN	MLP
RWGANs		
WGANs		
WGANs-GP		

Figure 3.7: Sample qualities at the final stage of training on CIFAR-10.





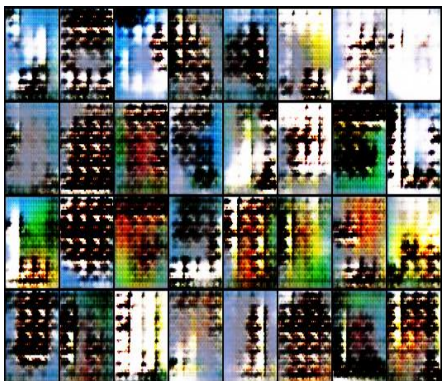
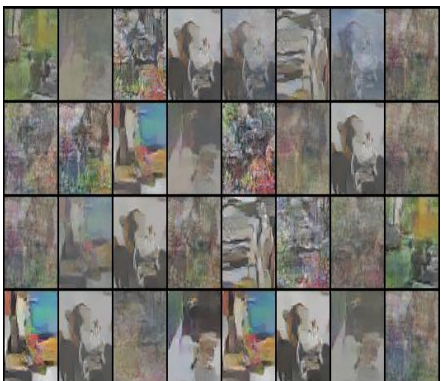
Method	$N = 1$	
	DCGAN	MLP
RWGANs		
WGANs		
WGANs-GP		

Figure 3.8: Sample qualities at the initial stage of training on ImageNet.

Method	$N = 25$	
	DCGAN	MLP
RWGANs		
WGANs		
WGANs-GP		

Figure 3.9: Sample qualities at the final stage of training on ImageNet.

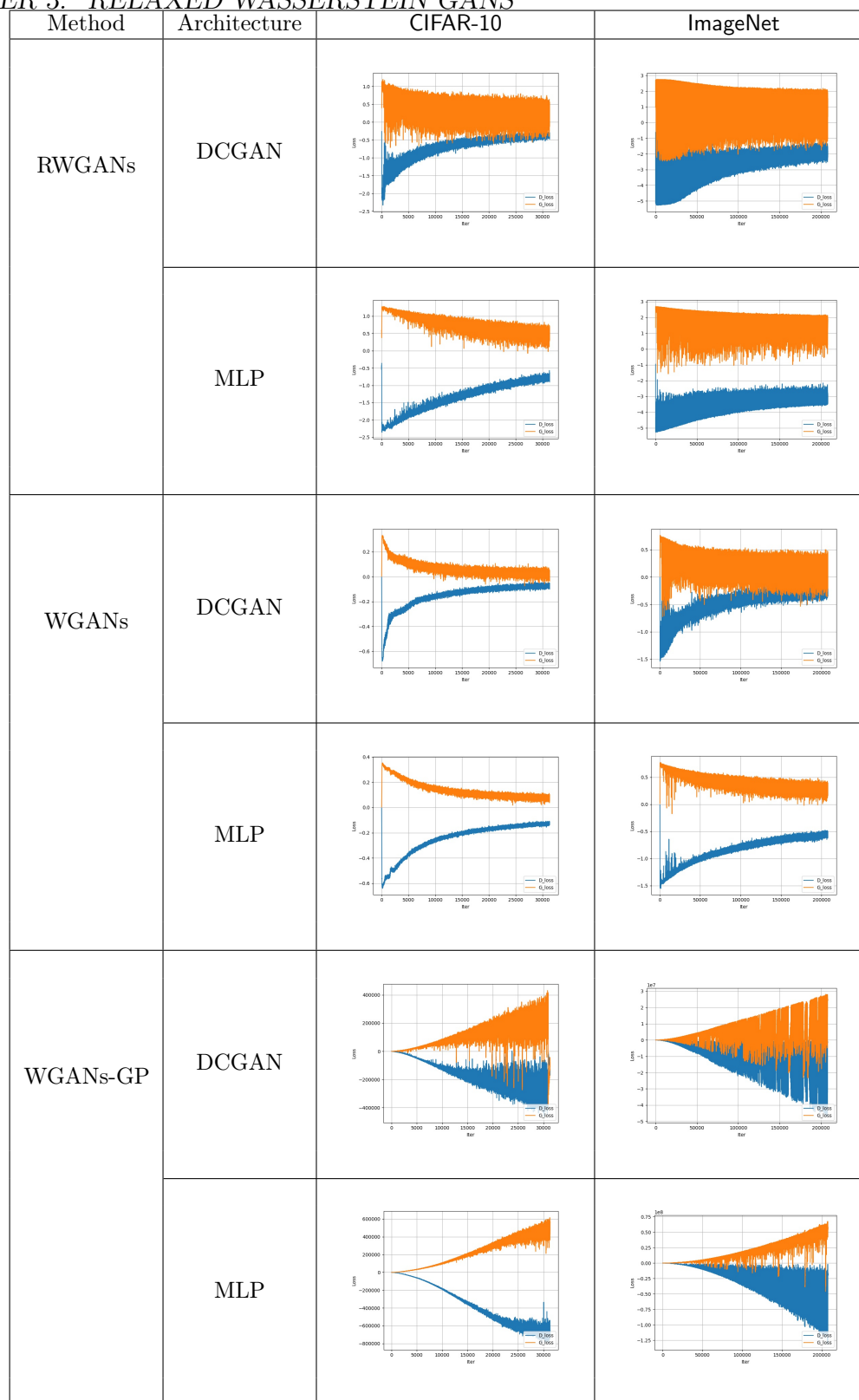


Figure 3.10: Training curves at different stages of training. DCGAN refers to the standard DCGAN generator and MLP refers to an ReLU-MLP with 4 hidden layers and 512 units at each layer. G_{loss} and D_{loss} refer to the loss in generative and discriminative nets. The loss in RWGANs is shown to converge consistently while the loss in WGANs-GP tends to diverge as the training progresses. WGANs achieves the lowest variance among the three methods.

Chapter 4

Properties of Bregman divergence and Choices of RW divergence

An important question in using RW divergence is the choice of the underlying convex function ϕ . In this chapter, we establish the connection between Fisher information and the Hessian of a convex function (Proposition 1), and derive the asymptotic distribution of Bregman divergences (Theorem 4.1.3). These theoretical results shed light from a variance stabilizing perspective on why Wasserstein- L^2 might not be a desirable choice of statistical divergence and how to select the convex function for RW.

In this chapter, our main contributions are as follows:

- We derive a weak convergence result using Bregman divergence in parametric distributions. The result describes precisely how the Hessian of the underlying convex function in Bregman divergence impacts the statistical properties of the divergence measure in the asymptotic setting.
- In the non-asymptotic setting, we prove concentration results using Bregman divergence between the true discrete distribution and the empirical distributions. This allows the construction of ambiguity set in robust optimization.

4.1 More Properties of Bregman Divergence

In Section 2.1, we introduce Bregman divergence and some of its properties. In this section, we will discuss several theorems that motivate our discussion later. Let $p = (p_1, \dots, p_m) \in \mathbb{R}^m$ be the probability distribution of a discrete random variable X , where $p_i = \mathbb{P}(X = a_i)$, $i \in \{1, 2, \dots, m\}$. Let $\hat{p}_n = (\hat{p}_{n,1}, \dots, \hat{p}_{n,m}) \in \mathbb{R}^m$ be the random vector denoting the empirical distribution of a sequence of iid random variables $\{X_i\}_{i=1}^n$, where each X_i has the same distribution as X . That is,

$$\hat{p}_n = \left(\frac{1}{n} \sum_{i=1}^n 1\{X_i = a_1\}, \dots, \frac{1}{n} \sum_{i=1}^n 1\{X_i = a_m\} \right).$$

Concentration of Bregman Divergence

We first establish that the Bregman divergence D_ϕ between the empirical distribution and the true distribution concentrates around the mean, where the rate can be expressed in terms of the gradient of the convex function ϕ .

Theorem 4.1.1. *Consider the random variable $Z = D_\phi(\hat{p}_n, p)$, the Bregman divergence between \hat{p}_n and p ,*

$$Z = D_\phi(\hat{p}_n, p) = \phi(\hat{p}_n) - \phi(p) - \langle \nabla \phi(p), \hat{p}_n - p \rangle,$$

where $\phi : [0, 1]^m \rightarrow \mathbb{R}$ is a strictly convex function. Then the following concentration inequality holds for all $\epsilon > 0$:

$$\mathbb{P}\{Z - \mathbb{E}[Z] \geq \epsilon\} \leq \exp\left(\frac{-n^2 \epsilon^2}{4dM_\phi}\right),$$

where $M_\phi = \max_{t \in \Delta^{m-1}} \|\nabla \phi(t)\|_2$, and Δ^{m-1} is the standard $(m-1)$ -simplex, which is the set $\{(t_1, t_2, \dots, t_m) \in \mathbb{R}^m \mid \sum_{i=1}^m t_i = 1, t_i \geq 0, \forall i\}$.

Proof. Let $(X_1, \dots, X_{i-1}, X_i, X_{i+1}, \dots, X_n)$ be iid random variables from distribution p . Define another sequence of random variables $(X_1, \dots, X_{i-1}, X'_i, X_{i+1}, \dots, X_n)$, in which only the i -th element in the sequence is different. Let the corresponding empirical distribution be \hat{p}'_n . Then

$$Z' = D_\phi(\hat{p}'_n, p) = \phi(\hat{p}'_n) - \phi(p) - \langle \nabla \phi(p), \hat{p}'_n - p \rangle.$$

The difference of Z and Z' is

$$Z' - Z = \phi(\hat{p}'_n) - \phi(\hat{p}_n) + \langle \nabla \phi(p), \hat{p}_n - \hat{p}'_n \rangle.$$

Notice that by construction, $\hat{p}_n - \hat{p}'_n$ is a vector with an element being $1/n$, an element being $-1/n$, and all other elements being zeros. Therefore by the Cauchy-Schwarz inequality,

$$\begin{aligned} |\langle \nabla \phi(p), \hat{p}_n - \hat{p}'_n \rangle| &\leq \|\nabla \phi(p)\|_2 \|\hat{p}_n - \hat{p}'_n\|_2 \\ &= \frac{\sqrt{2}}{n} \|\nabla \phi(p)\|_2 \leq \frac{\sqrt{2}}{n} M_\phi. \end{aligned}$$

Also by the Taylor's expansion,

$$\begin{aligned} \phi(\hat{p}'_n) - \phi(\hat{p}_n) &= \|\nabla \phi(\xi)\|_2 \|\hat{p}_n - \hat{p}'_n\|_2 \\ &\leq M_\phi \|\hat{p}_n - \hat{p}'_n\|_2 \\ &= \frac{\sqrt{2}}{n} M_\phi, \end{aligned}$$

where ξ is a random vector which is a convex combination of \hat{p}_n and \hat{p}'_n . Therefore by the triangle inequality,

$$\begin{aligned} |Z - Z'| &\leq |\langle \nabla \phi(p), \hat{p}_n - \hat{p}'_n \rangle| + |\phi(\hat{p}'_n) - \phi(\hat{p}_n)| \\ &\leq \frac{2\sqrt{2}}{n} M_\phi. \end{aligned}$$

Hence by the bounded difference inequality (Talagrand 1995),

$$\mathbb{P}\{Z - \mathbb{E}[Z] \geq \epsilon\} \leq \exp\left(\frac{-n^2\epsilon^2}{4dM_\phi}\right).$$

□

Notice that Bregman divergence is only convex with respect to its first argument, which in the previous case is \hat{p}_n . To construct a convex ambiguity region, we need to reverse the order of \hat{p} and p to make the unknown true distribution the first argument. Hence we also prove the following concentration inequality:

Theorem 4.1.2. *Consider the random variable $Y = D_\phi(p, \hat{p}_n)$, the Bregman divergence between p and \hat{p}_n :*

$$Y = D_\phi(p, \hat{p}_n) = \phi(p) - \phi(\hat{p}_n) - \langle \nabla \phi(\hat{p}_n), p - \hat{p}_n \rangle,$$

where $\phi : [0, 1]^m \rightarrow \mathbb{R}$ is a strictly convex function. Then we have the following concentration inequality for all $\epsilon > 0$:

$$\mathbb{P}(Y - \mathbb{E}Y \geq \epsilon) \leq \exp\left(-\frac{n^2\epsilon^2}{4d(M_\phi + L_\phi)^2}\right),$$

where L_ϕ is the Lipschitz constant of $\nabla \phi$, and $M_\phi = \max_{t \in \Delta^{m-1}} \|\nabla \phi(t)\|$. Δ^{m-1} is the standard $(d-1)$ -simplex, which is the set $\{(t_1, t_2, \dots, t_m) \in \mathbb{R}^m \mid \sum_{i=1}^m t_i = 1, t_i \geq 0, \forall i\}$.

Proof. Let $(X_1, \dots, X_{i-1}, X_i, X_{i+1}, \dots, X_n)$ be iid random variables from distribution p . Define another sequence of random variables $(X_1, \dots, X_{i-1}, X'_i, X_{i+1}, \dots, X_n)$, in which only the i -th element in the sequence is different. Let the corresponding empirical distribution be \hat{p}'_n . Then

$$Y' = D_\phi(p, \hat{p}'_n) = \phi(p) - \phi(\hat{p}'_n) - \langle \nabla \phi(\hat{p}'_n), p - \hat{p}'_n \rangle.$$

The difference of Y and Y' is

$$\begin{aligned} Y' - Y &= \phi(\hat{p}_n) - \phi(\hat{p}'_n) \\ &\quad + \langle \nabla \phi(\hat{p}_n), p - \hat{p}_n \rangle - \langle \nabla \phi(\hat{p}'_n), p - \hat{p}'_n \rangle. \end{aligned}$$

By the proof of Theorem 4.1.1,

$$\phi(\hat{p}_n) - \phi(\hat{p}'_n) \leq \frac{\sqrt{2}}{n} M_\phi.$$

Meanwhile

$$\begin{aligned} &\langle \nabla \phi(\hat{p}_n), p - \hat{p}_n \rangle - \langle \nabla \phi(\hat{p}'_n), p - \hat{p}'_n \rangle \\ &= \langle \nabla \phi(\hat{p}_n) - \nabla \phi(\hat{p}'_n), p \rangle \\ &\quad - \langle \nabla \phi(\hat{p}_n), \hat{p}_n \rangle + \langle \nabla \phi(\hat{p}'_n), \hat{p}'_n \rangle. \end{aligned}$$

Since $\nabla\phi$ is defined on the compact region $[0, 1]^m$, we can assume without loss of generality that it has Lipschitz constant L_ϕ . Then by the Cauchy-Schwarz inequality,

$$\begin{aligned} |\langle \nabla\phi(\hat{p}_n) - \nabla\phi(\hat{p}'_n), p \rangle| &\leq \|p\|_2 \|\nabla\phi(\hat{p}_n) - \nabla\phi(\hat{p}'_n)\|_2 \\ &\leq L_\phi \|\hat{p}_n - \hat{p}'_n\|_2 = \frac{\sqrt{2}}{n} L_\phi, \end{aligned}$$

and similarly

$$\begin{aligned} &| - \langle \nabla\phi(\hat{p}_n), \hat{p}_n \rangle + \langle \nabla\phi(\hat{p}'_n), \hat{p}'_n \rangle | \\ &= |\langle \nabla\phi(\hat{p}_n), \hat{p}'_n - \hat{p}_n \rangle + \langle \nabla\phi(\hat{p}'_n) - \nabla\phi(\hat{p}_n), \hat{p}'_n \rangle| \\ &\leq |\langle \nabla\phi(\hat{p}_n), \hat{p}'_n - \hat{p}_n \rangle| + |\langle \nabla\phi(\hat{p}'_n) - \nabla\phi(\hat{p}_n), \hat{p}'_n \rangle| \\ &\leq \frac{\sqrt{2}}{n} M_\phi + \frac{\sqrt{2}}{n} L_\phi. \end{aligned}$$

Therefore

$$|Y' - Y| \leq 2 \left(\frac{\sqrt{2}}{n} M_\phi + \frac{\sqrt{2}}{n} L_\phi \right).$$

By the bounded difference inequality,

$$\mathbb{P}(Y - \mathbb{E}Y \geq \epsilon) \leq \exp\left(-\frac{n^2\epsilon^2}{4d(M_\phi + L_\phi)^2}\right).$$

□

Weak Convergence of Bregman Divergence

In this section, we will show that in the asymptotic case, Bregman divergence between the true parameters of a distribution and the maximum likelihood estimator of the parameters will converge in distribution to a finite weighted sum of independent χ^2 distributed random variables. This result allows us to construct asymptotic ambiguity sets according to the quantiles of the asymptotic distribution.

Theorem 4.1.3. *Suppose there exists a family of probability distributions \mathbb{P}_θ parametrized by $\theta \in \Theta \subset \mathbb{R}^m$. Suppose we have iid data $\{X_i\}_{i=1}^n$, and $\hat{\theta}_n$ is the maximum likelihood estimator of θ . Then*

$$\lim_{n \rightarrow \infty} nD_\phi(\theta, \hat{\theta}_n) \xrightarrow{d} \frac{1}{2} \sum_{i=1}^r \beta_i Z_i^2,$$

where Z_i 's are independent standard Gaussian random variables, D_ϕ denotes the Bregman divergence characterized by ϕ , β_i 's are the non-zero eigenvalues of the matrix $H\Sigma$ and $r = \text{rank}(\Sigma^T H \Sigma)$, with H the Hessian of ϕ at θ and Σ the inverse Fisher information matrix.

Proof. First, write the Taylor expansion of ϕ around $\hat{\theta}_n$,

$$\begin{aligned}\phi(\theta) &= \phi(\hat{\theta}_n) + \langle \theta - \hat{\theta}_n, \nabla\phi(\hat{\theta}_n) \rangle \\ &\quad + \frac{1}{2}(\theta - \hat{\theta}_n)^T H(\hat{\theta}_n)(\theta - \hat{\theta}_n) + o(\|\theta - \hat{\theta}_n\|_2^2),\end{aligned}$$

where $H(\hat{\theta})$ is the Hessian of $\phi(x)$ at $x = \hat{\theta}$. Notice that by the properties of maximum likelihood estimators, as $n \rightarrow \infty$,

$$\sqrt{n}(\theta - \hat{\theta}_n) \xrightarrow{d} N(0, \mathcal{I}^{-1}) \stackrel{d}{=} N(0, \Sigma),$$

where

$$(\mathcal{I})_{ij} = -\mathbb{E} \frac{\partial^2 \log L}{\partial \theta_i \partial \theta_j}$$

is the Fisher information matrix of the underlying true distribution, with L being the likelihood function. Also,

$$H(\hat{\theta}_n) \rightarrow H(\theta)$$

in probability, and

$$n \cdot o(\|\theta - \hat{\theta}_n\|_2^2) \rightarrow 0$$

in probability. Therefore by the Slutsky's theorem,

$$\begin{aligned}nD_\phi(\theta, \hat{\theta}_n) &= n(\phi(\theta) - \phi(\hat{\theta}_n) - \langle \theta - \hat{\theta}_n, \nabla\phi(\hat{\theta}_n) \rangle) \\ &= \frac{1}{2}\sqrt{n}(\theta - \hat{\theta}_n)^T H\sqrt{n}(\theta - \hat{\theta}_n) \\ &\quad + n \cdot o(\|\theta - \hat{\theta}_n\|_2^2) \\ &\stackrel{d}{\rightarrow} \frac{1}{2}X^T H X,\end{aligned}$$

where $X \stackrel{d}{=} N(0, \Sigma)$. Let $S \in \mathbb{R}^{d \times s}$ be a square root of Σ . Since Σ and H are positive semidefinite, by spectral theorem, we can write $S^T H S = R^T \Lambda R$, where $\Lambda = \text{diag}(\beta_1, \dots, \beta_r)$, which is the diagonal matrix of non-zero eigenvalues of $S^T H S$, hence is also the diagonal matrix of non-zero eigenvalues of $H\Sigma$, $r = \text{rank}(\Sigma H \Sigma)$, and R is the matrix of corresponding orthonormal eigenvectors. Then

$$\begin{aligned}X^T H X &\stackrel{d}{=} (SY)^T H SY \stackrel{d}{=} Y^T R^T \Lambda R Y \\ &\stackrel{d}{=} Z^T \Lambda Z = \sum_{i=1}^r \beta_i Z_i^2,\end{aligned}$$

where Z_i are independent standard Gaussian random variables. Therefore, we have the quadratic form of Gaussian variables

$$\sqrt{n}(\theta - \hat{\theta}_n)^T H\sqrt{n}(\theta - \hat{\theta}_n) \stackrel{d}{=} \sum_{i=1}^r \beta_i Z_i^2.$$

This completes the proof. □

Remark 3. Even though Bregman divergence is asymmetric, $nD_\phi(\hat{p}_n, p)$ has the same asymptotic distribution as $nD_\phi(p, \hat{p}_n)$ by a similar proof.

Noting that \hat{p}_n is the maximum likelihood estimator of p , we immediately arrive at the following corollary.

Corollary 4.1.3.1. For a discrete distribution $p = (p_1, \dots, p_m)$ and the empirical distribution $\hat{p}_n = (\hat{p}_{n,1}, \dots, \hat{p}_{n,m})$ generated from n iid samples, we have

$$\lim_{n \rightarrow \infty} nD_\phi(p, \hat{p}_n) \xrightarrow{d} \frac{1}{2} \sum_{i=1}^r \beta_i Z_i^2,$$

where Z_i are independent standard Gaussian random variables, $r = \text{rank}(\Sigma^T H \Sigma)$, H is the Hessian of ϕ , Σ is the inverse Fisher information matrix, and β_1, \dots, β_r are the nonzero eigenvalues of $H\Sigma$.

4.2 Choices of ϕ in Relaxed Wasserstein divergence

While RW divergence provides the flexibility of choosing the underlying Bregman divergence, in practice one would like to have a principled way of determining this choice. The following theoretical results shed light on how to choose an appropriate convex function ϕ in Bregman divergence D_ϕ .

Our result connects Fisher information of a distribution of an exponential family and the Hessian of ϕ .

Proposition 1. Suppose $X \sim \mathbb{P}_\theta$ belongs to a regular exponential family. Let $\mu = \mathbb{E}(X)$, ψ be the cumulant function, and ϕ be the convex conjugate of ψ . Let

$$(\mathcal{I})_{ij} = -\mathbb{E} \frac{\partial^2 \log L}{\partial \theta_i \partial \theta_j} \tag{4.1}$$

be the Fisher information matrix of the underlying true distribution, with L being the likelihood function. Assume that ψ is three-time differentiable. Then

$$\mathcal{I}(\mu) = \mathbb{E} [\nabla_\mu^2 D_\phi(x, \mu)] = \nabla^2 \phi(\mu). \tag{4.2}$$

Proof. Equation (4.1) follows directly from the representation $p_\theta(x) = \exp(-D_\phi(x, \mu) - g_\phi(x))$. Equation (4.2) follows from a straightforward calculation,

$$\begin{aligned} \mathbb{E} [\nabla_\mu^2 D_\phi(x, \mu)] &= \mathbb{E} [\nabla_\mu^2 [\phi(x) - \phi(\mu) - \nabla \phi(\mu)^T (x - \mu)]] \\ &= \mathbb{E} [\nabla_\mu [-\nabla^2 \phi(\mu)(x - \mu)]] = \mathbb{E} [\nabla_\mu^2 \phi(\mu)] = \phi''(\mu). \end{aligned}$$

□

Recall that Theorem 4.1.3 shows that asymptotically, Bregman divergence between the true parameters and the corresponding maximum likelihood estimator of the parameters will converge in distribution to a finite weighted sum of independent χ^2 distributed random variables.

To see how Theorem 4.1.3 and Proposition 1 shed light on the choice of ϕ , let us first consider the squared loss $D_\phi(x, y) = \|x - y\|^2$. In this case, the corresponding ϕ is $\|x\|_2^2$ and the Hessian is $H = 2I$, so the weights in the weighted sum of χ_1^2 random variables in Theorem 4.1.3 are determined purely from the eigenvalues of the inverse Fisher information matrix Σ , which is the negative inverse of Hessian of the likelihood function. In other words, the convergence behavior of $nD_\phi(\theta, \hat{\theta}_n)$ is purely determined from the curvature of the likelihood surface at θ . If the likelihood surface is close to being flat at θ in certain directions, some of the eigenvalues of Σ will be undesirably large, resulting in a large asymptotic variance for $nD_\phi(\theta, \hat{\theta})$. This suggests that the squared loss function and hence Wasserstein- L^2 might not be a suitable choice as a divergence measure when the underlying likelihood function is likely to be flat at the true parameter θ .

Moreover, in light of Theorem 4.1.3, H can be used as a tool to stabilize the asymptotic variations of $nD_\phi(\theta, \hat{\theta}_n)$. A potential choice of H is Σ^{-1} , the Fisher information matrix of the likelihood function. Then $H\Sigma = I$, so all the associated eigenvalues β_i 's are ones and the resulting asymptotic variance is always $r/2$, independent of the curvature of the underlying likelihood surface. To ensure that $H = \Sigma^{-1}$, if the underlying likelihood function is from an exponential family, ϕ can be simply chosen to be the associated Bregman divergence by Proposition 1. Note that with this choice $nD_\phi(\theta, \hat{\theta}_n)$ is equivalent to the classical likelihood ratio statistic. In a more general setting, if a reasonable estimate of the Fisher information matrix at θ is available, say $\hat{\Sigma}^{-1}$, a reasonable choice of Bregman divergence is the Mahalanobis distance $D_\phi(x, y) = (x - y)^T \hat{\Sigma}^{-1} (x - y)$, provided that the objective is to stabilize the asymptotic variance of $nD_\phi(\theta, \hat{\theta}_n)$. Indeed, the corresponding ϕ is $\phi(x) = x^T \hat{\Sigma}^{-1} x$ and the Hessian is $\hat{\Sigma}^{-1}$, so the matrix $H\Sigma$ in Theorem 4.1.3 is close to being the identity matrix.

Chapter 5

Applications

In this chapter, we cover three examples using techniques discussed in previous chapters: distributionally robust optimization problems (DRO) for inventory and portfolio management, variational autoencoders for metagenomic binning, and generative adversarial networks for financial data simulation.

5.1 Distributionally Robust Optimization

Solutions to traditional optimization problems are usually sensitive to the model parameters, which is a major drawback. Robust optimization solves this issue by formulating problems under appropriate uncertainty sets for the model parameters and/or for the solutions against a certain measure of robustness. For instance, the constraint $f(x, z) \leq 0$ for all $z \in \mathcal{C}$ is robust for the decision variable $x \in \mathbb{R}$. $z \in \mathbb{R}$ is an uncertain parameter, $\mathcal{C} \subset \mathbb{R}$ is the uncertainty set and $f : \mathbb{R} \rightarrow \mathbb{R}$ is a constraint function. For more details, please refer to Ben-Tal and A. Nemirovski (1998) and Ben-Tal, El Ghaoui, and A. Nemirovski (2009) for the tractability of such formulations.

As another example, instead of requiring that the constraint be satisfied at all time, tractable uncertainty sets can be formulated in terms of chance constraints and expectation constraints under a given distribution \mathbb{P} (Jiang and Guan 2016) such that the constraints are satisfied with certain probability.

However, in most data-driven research, the distribution \mathbb{P} itself is usually unknown. The concept of ambiguity sets is introduced in Scarf (1957), and then explored in, for example, Delage and Ye (2010), Ghaoui, Oks, and Oustry (2003), and Bayraksan and Love (2015). The key idea of Distributionally Robust Optimization (DRO) is as follows: instead of optimizing under one particular distribution and under a deterministic set, it formulates optimization problems with a set of possible distributions, under the concept of ambiguity sets. The ambiguity set contains distributions that are not far away from the nominal distribution, measured by the divergence function.

Specifically, one could consider minimizing the expected loss as follows,

$$\min_{X \in \mathcal{X}} \max_{\mathbb{P} \in \mathcal{P}} \mathbb{E}_{\mathbb{P}}[h(X; \xi)],$$

where X is the decision variable, allowed to vary inside the feasible region \mathcal{X} , and the random element ξ follows distribution $\mathbb{P} \in \mathcal{P}$, with \mathcal{P} being the ambiguity set and h the loss function.

In the data driven setting where we have iid samples $\{\xi_i\}_{i=1}^n$ drawn from \mathbb{P} , the ambiguity set \mathcal{P} can be constructed so that it contains all distributions that are within a certain divergence from the empirical distribution, where the radius of the ambiguity set is large enough so that it contains \mathbb{P} with high probability. Alternative methods to construct ambiguity sets use moment constraints under $\mathbb{P} \in \mathcal{P}$, where \mathcal{P} consists of all probability distributions with first order and second order moments matching the sample moments. Again, the key is to define and measure the difference between various distributions.

In this section, we will discuss cases where we use the Bregman divergence and the Relaxed Wasserstein divergence to construct ambiguity sets.

Related Work

Comparing probability distributions has been a recurring theme in many research areas of machine learning. In distribution learning, for example, one is interested in approximating the true distribution by an element in a predetermined class of probability distributions, and this element is chosen based on the observed data. Such choices rely on the divergence used in comparing distributions. While there is an abundance in statistical divergences, there is no consensus about the "ideal" way to measure the difference between distributions.

In this regard, various choices of divergence functions have been discussed in the literature of distributionally robust optimization, for example, KL and f -divergences (Namkoong and Duchi 2016; Van Parys, Esfahani, and Kuhn 2017) and Wasserstein distances (Esfahani and Kuhn 2018; Shafieezadeh-Abadeh, Esfahani, and Kuhn 2015; Powell 2016; Gao and Kleywegt 2016; Blanchet, Lin Chen, and X. Y. Zhou 2018).

DRO with KL Divergence. In Hu and Hong (2013), they formulate a robust optimization problem in terms of a KL divergence constraint and show that the problem can be converted into a convex optimization problem which can be solved analytically. In Jiang and Guan (2016), they show that chance constraints with KL divergence ambiguity sets can be reformulated into a traditional chance constraint problem with different risk levels.

DRO with L_p -Wasserstein Distance. In Esfahani and Kuhn (2018), they propose the use of L_1 -Wasserstein ambiguity set. They show that Wasserstein ambiguity sets provide a better out-of-sample guarantee than the KL divergence, because a continuous \mathbb{P} will always be outside the KL divergence ball centered at the empirical distribution $\hat{\mathbb{P}}_n$, which is discrete, whereas the Wasserstein ball contains continuous as well as discrete distributions. They also show that the robust optimization problem, under some mild conditions, can be converted

into a finite-dimensional convex programming problem, solvable in polynomial time. In Shafieezadeh-Abadeh, Esfahani, and Kuhn (2015), they use Wasserstein ambiguity set for distributionally robust logistic regression. Specifically they study $\inf_{\beta} \sup_{\mathbb{P} \in \mathcal{P}} \mathbb{E}_{\mathbb{P}}[l_{\beta}(x, y)]$, where $l_{\beta}(x, y)$ is the logloss function with parameter β . They show that this problem has a tractable convex reformulation and provide confidence interval for the objective function, which is the out of sample performance guarantee. In Powell (2016), they use the L_1 -Wasserstein ball as the ambiguity set. They show that the candidate probability distributions in the ball can be reduced to a subset whose elements can be described using extreme/exposed points of the set, hence a tractable reformulation of the original problem becomes possible. In Gao and Kleywegt (2016), they consider the L_p -Wasserstein ball for $p \geq 1$, and give necessary and sufficient conditions for the worst-case distributions to exist. In Fournier and Guillin (2015), they inspect the convergence rate of the empirical distribution to the true distribution under Wasserstein distance.

DRO with Bregman divergence

In this section, we apply Bregman ambiguity sets to two types of problems: Basic Distributionally Robust Problem, and the Distributionally Robust Optimization problem, using two specific Bregman divergences: KL divergence and the Euclidean distance. Suppose one chooses the divergence between probability distributions to be $d(\mathbb{P}, \mathbb{Q})$, where \mathbb{P} and \mathbb{Q} are probability measures defined on the set $\mathcal{X} \subset \mathbb{R}^n$. Let \mathcal{M}_+ denotes the set of all probability distributions defined over the set \mathcal{X} . Then the ambiguity set \mathcal{P} can be defined as a ball centered at the nominal distribution \mathbb{Q} :

$$\mathcal{P} = \{\mathbb{P} \in \mathcal{M}_+ : d(\mathbb{P}, \mathbb{Q}) \leq \delta\}.$$

The nominal distribution \mathbb{Q} may come from prior knowledge of the model, or directly from data. In the data-driven setting where we are given iid samples $\{X_i\}_{i=1}^n$, the nominal distribution \mathbb{Q} is chosen to be the empirical distribution $\hat{\mathbb{P}}_n$.

- When the sample size n is large (relative to d), one can appeal to the asymptotic distribution of $D(p, \hat{p}_n)$ to construct an ambiguity set using Theorem 4.1.3.1. More specifically, an ambiguity set can be constructed as follows:

$$\mathcal{P} = \{p : D_{\phi}(p, \hat{p}_n) \leq \frac{1}{2n} F^{-1}(\alpha)\},$$

where $F^{-1}(\alpha)$ is the quantile function of $\sum_{i=1}^r \beta_i Z_i^2$, which is a weighted sum of independent χ^2 random variables with one degree of freedom. This quantile can be approximated via a Monte Carlo approximation. For a large K (say $K = 10000$), one can simulate rK independent standard normal random variables $Z_{1,1}, \dots, Z_{1,r}, Z_{2,1}, \dots, Z_{2,r}, \dots, Z_{K,1}, \dots, Z_{K,r}$, and compute $R_j = \sum_{i=1}^r \beta_i Z_{i,j}^2$ for each $j = 1, \dots, K$. Then one can use take the α -th empirical quantile of (R_1, \dots, R_K) as an approximation to $F^{-1}(\alpha)$. Note that \mathcal{P} is convex since Bregman divergence is convex with respect to the first argument.

- When the sample size n is of moderate size or small, one must appeal to concentration results to obtain a valid ambiguity set. In order to apply Theorem 4.1.1 or Theorem 4.1.2 for the construction of the ambiguity set, we have to first derive an upper bound for $\mathbb{E}D_\phi(p, \hat{p}_n)$ or $\mathbb{E}D_\phi(\hat{p}_n, p)$, respectively.

For $\mathbb{E}D_\phi(p, \hat{p}_n)$, clearly

$$\begin{aligned}
\mathbb{E}D_\phi(p, \hat{p}_n) &= \mathbb{E}[\phi(p) - \phi(\hat{p}_n) - \langle \nabla\phi(\hat{p}_n), p - \hat{p}_n \rangle] \\
&= \mathbb{E}[\phi(p) - \phi(\hat{p}_n)] - \mathbb{E}[\langle \nabla\phi(p) - \nabla\phi(\hat{p}_n), p - \hat{p}_n \rangle] \\
&\leq M_\phi \sqrt{\sum_{i=1}^d \frac{p_i(1-p_i)}{n}} + L_\phi \mathbb{E}\|p - \hat{p}_n\|_2^2 \\
&= M_\phi \sqrt{\sum_{i=1}^d \frac{p_i(1-p_i)}{n}} + L_\phi \sum_{i=1}^d \frac{p_i(1-p_i)}{n} \\
&= M_\phi \sqrt{\frac{d}{4n}} + L_\phi \frac{d}{4n},
\end{aligned}$$

where the inequality is by the Cauchy-Schwarz inequality and the Taylor's theorem.

Similarly, for $\mathbb{E}D_\phi(\hat{p}_n, p)$,

$$\begin{aligned}
\mathbb{E}[D_\phi(\hat{p}_n, p)] &= \mathbb{E}[\phi(\hat{p}_n) - \phi(p)] \\
&= \mathbb{E}[\langle \nabla\phi(\xi), \hat{p} - p \rangle] \\
&\leq \mathbb{E}[\|\nabla\phi(\xi)\| \|\hat{p} - p\|_2] \\
&\leq M_\phi \mathbb{E}[\|\hat{p} - p\|_2] \\
&\leq M_\phi \sqrt{\mathbb{E}[\|\hat{p} - p\|_2^2]} \\
&= M_\phi \sqrt{\sum_{i=1}^d \frac{p_i(1-p_i)}{n}} \\
&\leq M_\phi \sqrt{\frac{d}{4n}},
\end{aligned}$$

where ξ is between \hat{p} and p , the first inequality is by Cauchy-Schwarz, and the third inequality is by the Jensen's inequality.

- As described immediately after the proof of Theorem 4.1.1, the resulting ambiguity set might be intractable to be computed because of its potential nonconvex nature. On the other hand, Theorem 4.1.2 results in a convex ambiguity set, which is the Bregman ball of \hat{p}_n with radius $M_\phi \sqrt{\frac{d}{4n}} + L_\phi \left(\frac{d}{4n}\right) + \epsilon$:

$$\left\{ p : D_\phi(p, \hat{p}_n) \leq M_\phi \sqrt{\frac{d}{4n}} + L_\phi \left(\frac{d}{4n}\right) + \epsilon \right\}.$$

- Consider the problem that involves two parametrized distributions p_β and p_{β_0} :

$$\begin{aligned} \max_{p_\beta} / \min_{p_\beta} \sum_{k=1}^K \ell(\beta, \beta_0) \\ \text{s.t. } D_\phi(\beta, \beta_0) \leq \delta \end{aligned}$$

By Lagrangian duality, we know that for every $\delta > 0$, there exists a corresponding $\lambda(\delta) > 0$ such that the following problem will yield the same solution:

$$\max_{p_\beta} / \min_{p_\beta} \sum_{k=1}^K \ell(\beta, \beta_0) + \lambda(\delta) D_\phi(\beta, \beta_0)$$

This can be seen as an regression/classification problem in statistical learning, with $D_\phi(\beta, \beta_0)$ as the regularization term, penalizing β 's that are far from the nominal β_0 measured by the Bregman divergence.

In the coming sections, we will give some examples where we can use the Bregman ambiguity sets.

Basic Distributionally Robust Problem with KL ambiguity set

Consider the following setting: $\{h_i\}_{i=1}^m$ is a sequence of real numbers representing the cost incurred in scenario i . $\{p_{0,i}\}_{i=1}^m$ is the baseline model where $p_{0,i}$ represents the probability that scenario i occurs. In the data-driven case, we can set $p_{0,i} = \hat{p}_i$, which is the empirical probability of scenario i . Our objective is to find a robust confidence interval of the expected cost $\mathbb{E}_p[h]$. Mathematically the problem can be formulated as 5.1:

$$\begin{aligned} \max_p / \min_p \sum_{i=1}^m p_i h_i \\ \text{s.t. } D_\phi(p, p_0) \leq \delta \\ \sum_{i=1}^m p_i = 1 \\ p_i \geq 0 \text{ for } 1 \leq i \leq m \end{aligned} \tag{5.1}$$

Example 1. Choose D_ϕ to be the KL divergence (called Basic Distributionally Robust (BDR) Problem in Blanchet, Lam, et al. (2017))

$$D_\phi(p, p_0) = \sum_{i=1}^m p_m \log \left(\frac{p_i}{p_{0,i}} \right).$$

In this case, the optimal solution ((Blanchet, Lam, et al. 2017)) is

$$p_i = p_{0,i} \exp(\theta h_i - \psi),$$

where $\theta = 1/\lambda_1$, and $\psi = \lambda_2/\lambda_1 + 1$ satisfying

$$\theta \frac{\sum_{i=1}^m p_{0,i} \exp(\theta h_i) h_i}{\sum_{i=1}^m p_{0,i} \exp(\theta h_i)} - \log \left(\sum_{i=1}^m p_{0,i} \exp(\theta h_i) \right).$$

In general, since the constraints are convex with respect to p_i , the objective function is linear and $p = p_0$ is an interior point, Slater's condition is satisfied and the KKT condition is necessary to find the optimal solution. The Lagrangian for this problem is

$$L(p, \lambda) = - \sum_i p_i h_i + \lambda_1 (D_\phi(p, p_0) - \delta) + \lambda_2 (\sum_i p_i - 1).$$

In cases where this formulation is too complex to solve analytically, the Mirror Descent algorithm can be applied to 5.1. Notice that a single update step of the mirror descent algorithm applied to the problem without the ambiguity constraint with the prox function being $\phi(\cdot)$ is equivalent to solving the problem itself. The algorithm update is listed as Algorithm 4.

Algorithm 4 Mirror Descent

for $t = 1, 2, \dots, T$ **do**

$$g^{(t)} = (h_1, \dots, h_m) \in \partial f(p^{(t)})$$

$$p^{(t+1)} = \arg \min_{p \in \Delta_m} \{f(p^{(t)}) + g_t^T(p - p^{(t)}) + \frac{1}{\alpha_t} D_\phi(p, p^{(t)})\}$$

end for

Distributionally Robust Optimization with L2 ambiguity set

Consider the following problem with a parameter/decision variable $x \in \mathcal{X}$ where $\mathcal{X} \subseteq \mathbb{R}^d$ is a convex compact feasible region for the decision variables. This is called the Distributionally Robust Optimization problem (DRO). For a sequence of convex smooth cost functions $h_k(\cdot) : \mathcal{X} \rightarrow \mathbb{R}$, we want to provide a robust confidence interval of the expected objective value $\mathbb{E}_p[h(x)]$, using the Bregman ambiguity set. Formally,

$$\begin{aligned} \max_p / \min_p \min_{x \in \mathcal{X}} \sum_{i=1}^m p_i h_i(x) \\ \text{s.t. } D_\phi(p, p_0) \leq \delta \\ \sum_{i=1}^m p_i = 1, \\ p_i \geq 0 \text{ for } i \geq 1 \end{aligned} \tag{5.2}$$

For the max-min problem, since the feasible region for p is a Bregman ball, which is a compact subset of \mathbb{R}^m , by Sion's minimax theorem, we can switch the order of min and max so that 5.2 is reformulated as two subproblems: min-max (5.3) :

$$\begin{aligned} \min_{x \in \mathcal{X}} \max_p \sum_{i=1}^m p_i h_i(x) \\ \text{s.t. } D_\phi(p, p_0) \leq \delta \\ \sum_{i=1}^m p_i = 1, p_i \geq 0 \text{ for } i \geq 1 \end{aligned} \quad (5.3)$$

and min-min (5.4)

$$\begin{aligned} \min_{x \in \mathcal{X}} \min_p \sum_{i=1}^m p_i h_i(x) \\ \text{s.t. } D_\phi(p, p_0) \leq \delta \\ \sum_{i=1}^m p_i = 1, p_i \geq 0 \text{ for } i \geq 1 \end{aligned} \quad (5.4)$$

Example 2. *The special case of applying KL divergence to this problem has been considered in Lam and E. Zhou (2015). Consider instead*

$$D_\phi(p, p_0) = \sum_{i=1}^m (p_i - p_{0,i})^2$$

corresponding to $\phi(x) = \frac{1}{2} \|x\|_2^2$. This example has an analytical solution. First, it can be shown that the nonnegativity constraints can be ignored for sufficiently small δ . Then using the KKT conditions, the solution to the maximization problem can be shown to be

$$\begin{aligned} \lambda_1 &= \sqrt{\frac{\sum_{i=1}^m (h_i - \bar{h})^2}{2\delta}} \\ \lambda_2 &= \frac{1}{m} \sum_{i=1}^m h_i := \bar{h} \\ p_i &= \frac{h_i - \lambda_2}{\lambda_1} + p_{0,i} \end{aligned}$$

and the solution to the minimization problem is

$$\begin{aligned}\lambda_1 &= \sqrt{\frac{\sum_{i=1}^m (h_i + \bar{h})^2}{2\delta}} \\ \lambda_2 &= \frac{1}{m} \sum_{i=1}^m h_i := \bar{h} \\ p_i &= \frac{-h_i - \lambda_2}{\lambda_1} + p_{0,i}\end{aligned}$$

The shape of this ambiguity set in terms of the upper and lower bound of the objective is illustrated in Figure 5.1:

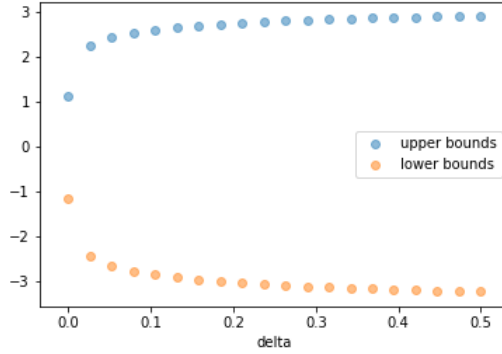


Figure 5.1: Shape of the upper and lower bounds of the objective value using an L^2 ambiguity set.

In general, if the inner problem is analytically solvable, then the whole problem would just be minimizing over $x \in \mathcal{X}$, which turns to be a traditional optimization problem. If the problem has a structure that is too complex to solve analytically, then the min-min problem can be solved by the coordinate descent algorithm.

To solve the min-max problem, we can apply the Mirror Prox algorithm (A. Nemirovski 2004; Juditsky, A. Nemirovski, and Tauvel 2011). The algorithm is suitable for a saddle-point problem, which in our case would be maximizing a concave function over p and minimizing a convex function over x .

Define the prox mapping as $\Pi_{\mathcal{Z}}(x) = \arg \min_{z \in \mathcal{Z}} [\langle x, z \rangle + D_\phi(z, z')]$, where $\mathcal{Z} = \mathcal{X} \times \mathcal{B}_{D_\phi}(\delta)$. The procedure for this Mirror Prox algorithm is shown in Algorithm 5.

Here α_t is the stepsize for step t , $g_t = (\nabla_x f(x, p), -\nabla_p f(x, p))$ is the $(d + K)$ dimensional vector of gradients at step t of $f(x, p) = \sum_{i=1}^m p_i h_i(x)$. $z_t = (x_t, p_t)$ is the vector of decision variables at iteration t . This algorithm achieves $O(\frac{1}{T})$ convergence rate when we have h_i smooth for all i , and $O(\frac{1}{\sqrt{T}})$ convergence rate when we drop the smoothness assumption.

Algorithm 5 Mirror Prox algorithm for DRO

```

Initialize  $z_0$ 
for  $t = 1, 2, \dots, T_{\max}$  do
     $a_t = \Pi_{z_t}(\alpha_t g(z_t))$ 
     $z_{t+1} = \Pi_{z_t}(\alpha_t g(a_t))$ 
end for

```

Robust optimization problems with linear constraints

Given $a \in \mathbb{R}^n$, $B \in \mathbb{R}^{n \times m}$, $\beta \in \mathbb{R}$, $C \in \mathbb{R}^{k \times m}$, $\rho \geq 0$, the following theorem shows that the Bregman ambiguity sets have a similar duality result to the ambiguity sets constructed using ϕ -divergences with linear constraints (see for example (Ben-Tal, Den Hertog, et al. 2013)).

Theorem 5.1.1. *Suppose we have the ambiguity set $\mathcal{U} = \{p \in \mathbb{R}^m | p \geq 0, Cp \leq d, D_\phi(p, q) \leq \rho\}$ and the robust linear constraint $(a + Bp)^T x \leq \beta, \forall p \in \mathcal{U}$. Assume that $q \in \mathcal{U}$, then $x \in \mathbb{R}^m$ is feasible if there exists non-negative Lagrangian multipliers $\eta \in \mathbb{R}^k$ and $\lambda \in \mathbb{R}$ such that*

$$a^T x + d^T \eta + \rho \lambda + \lambda \sum_{i=1}^m [\phi(q_i) - \nabla \phi(q_i) q_i] + \sum_{i=1}^m \phi^* \left(\frac{b_i^T x - c_i^T \eta}{\lambda} + \nabla \phi(q_i) \right) \leq \beta.$$

Proof. The feasibility of x is equivalent to

$$\begin{aligned} \beta &\geq \max_p \{(a + Bp)^T x | p \in \mathcal{U}\} \\ &= \max_{p \geq 0} \{(a + Bp)^T x | Cp \leq d, \sum_{i=1}^m [\phi(p_i) - \phi(q_i) - \nabla \phi(q_i)(p_i - q_i)] \leq \rho\}. \end{aligned}$$

The Lagrange function for the maximization problem above is

$$L(p, \lambda, \eta) = (a + Bp)^T x + \rho \lambda - \lambda \sum_{i=1}^m [\phi(p_i) - \phi(q_i) - \nabla \phi(q_i)(p_i - q_i)] + \eta^T (d - Cp).$$

The dual objective function is

$$g(\lambda, \eta) = \max_{p \geq 0} L(p, \lambda, \eta).$$

Since \mathcal{U} is regular, the strong duality holds. We can write

$$\begin{aligned}
g(\lambda, \eta) &= a^T x + d^T \eta + \rho\lambda + \lambda \sum_{i=1}^m [\phi(q_i) - \nabla\phi(q_i)q_i] + \max_{p \geq 0} \sum_{i=1}^m [p_i(b_i^T x) - p_i(c_i^T \eta) - \lambda\phi(p_i) + \lambda p_i \nabla\phi(q_i)] \\
&= a^T x + d^T \eta + \rho\lambda + \lambda \sum_{i=1}^m [\phi(q_i) - \nabla\phi(q_i)q_i] + \max_{p \geq 0} \sum_{i=1}^m [p_i(b_i^T x - c_i^T \eta + \lambda \nabla\phi(q_i)) - \lambda\phi(p_i)] \\
&= a^T x + d^T \eta + \rho\lambda + \lambda \sum_{i=1}^m [\phi(q_i) - \nabla\phi(q_i)q_i] + \sum_{i=1}^m (\lambda\phi)^* (b_i^T x - c_i^T \eta + \lambda \nabla\phi(q_i)) \\
&= a^T x + d^T \eta + \rho\lambda + \lambda \sum_{i=1}^m [\phi(q_i) - \nabla\phi(q_i)q_i] + \sum_{i=1}^m \phi^* \left(\frac{b_i^T x - c_i^T \eta}{\lambda} + \nabla\phi(q_i) \right).
\end{aligned}$$

□

Then the following corollary holds:

Corollary 5.1.1.1. *Given $f : \mathbb{R}^n \rightarrow \mathbb{R}^k$. Suppose we have the ambiguity set $\mathcal{U} = \{p \in \mathbb{R}^m | p \geq 0, Cp \leq d, D_\phi(p, q) \leq \rho\}$ and the robust linear constraint $(a + Bp)^T f(x) \leq \beta, \forall p \in \mathcal{U}$. Assume that $q \in \mathcal{U}$, then $x \in \mathbb{R}^m$ is feasible if there exists non-negative Lagrangian multipliers $\eta \in \mathbb{R}^k$ and $\lambda \in \mathbb{R}$ such that*

$$a^T f(x) + d^T \eta + \rho\lambda + \lambda \sum_{i=1}^m [\phi(q_i) - \nabla\phi(q_i)q_i] + \sum_{i=1}^m \phi^* \left(\frac{b_i^T f(x) - c_i^T \eta}{\lambda} + \nabla\phi(q_i) \right) \leq \beta.$$

Observing this corollary, let us reconsider the min-max problem (5.2). Assuming that the cost functions $h_i(x)$ are concave in x , it can be formulated as

$$\max_{x \in \mathcal{X}} \min_{p \in \mathcal{U}} \sum_{i=1}^m p_i h_i(x).$$

By adding a dummy variable z which represents the cost in the worst case, this problem is equivalent to the reformulation

$$\max_{z, x \in \mathcal{X}} \left\{ z \mid \sum_{i=1}^m p_i h_i(x) \geq z, \forall p \in \mathcal{U} \right\}.$$

Then by Corollary 5.1.1.1, 5.2 is equivalent to

$$\max_{x \in \mathcal{X}, \lambda \geq 0, \eta} \left\{ -\eta - \rho\lambda - \lambda \sum_{i=1}^m [\phi(q_i) - \nabla\phi(q_i)q_i] - \sum_{i=1}^m \phi^* \left(\frac{-h_i(x) - \eta}{\lambda} + \nabla\phi(q_i) \right) \right\}. \quad (5.5)$$

This shows that the robust counterpart of a DRO with linear constraints can be reduced to a ordinary convex (concave) optimization problem.

An Example in Inventory Management

In this section, we consider the classical example of the Newsvendor problem. In this setting, a merchant, faced with random demand D , would like to decide the optimal inventory level Q . Suppose that the profit of selling one unit of goods is u , the acquisition cost is c , the salvage value is s , and the cost of lost sales is l . Then the value function of the merchant is:

$$f(Q) = u \min(D, S) - cS + s(Q - D)_+ + l(D - Q)_+.$$

If $\mathbb{P}(D = d_i) = p_i, i = 1 \dots m$, then

$$h_i(Q) = u \min(d_i, S) - cS + s(Q - d_i)_+ + l(d_i - Q)_+.$$

If the merchant is trying to maximize the worst case revenue, then the optimization problem is:

$$\max_Q \min_{p \in \mathcal{U}} \sum_{i=1}^m p_i h_i(Q).$$

Then by Equation 5.5, this problem is equivalent to

$$\min_{Q, \lambda \geq 0, \eta} \left\{ \eta + \rho\lambda + \lambda \sum_{i=1}^m [\phi(q_i) - \nabla\phi(q_i)q_i] + \sum_{i=1}^m \phi^* \left(\frac{-h_i(x) - \eta}{\lambda} + \nabla\phi(q_i) \right) \right\}.$$

To add a linear constraint to this problem, let's assume that there are in total K items available for stock. The items face independent demands $D^{(k)}$, but the merchant has budget constraint such that the total money invested in all items should not exceed β . Therefore the whole problem can be formulated into a distributionally robust multi-item Newsvendor problem as follows:

$$\begin{aligned} \max_Q \min_{p^{(k)} \in \mathcal{U}} & \sum_{k=1}^K \sum_{i=1}^m p_i^{(k)} h_i^{(k)}(Q^{(k)}) \\ \text{s.t.} & \sum_{k=1}^K c^{(k)} Q^{(k)} \leq \beta. \end{aligned}$$

Then by previous discussions, this problem can be transformed into:

$$\begin{aligned} \max & \|z\|_\infty \\ \text{s.t.} & -c_k Q^{(k)} - \eta_k - \lambda_k \rho - \lambda_k \sum_{i=1}^m [\phi(q_i^{(k)}) - \nabla\phi(q_i^{(k)})q_i^{(k)}] \\ & - \sum_{i=1}^m \phi^* \left(\frac{-h_i^{(k)}(x) - \eta_k}{\lambda_k} + \nabla\phi(q_i^{(k)}) \right) \geq z_k, \forall k \\ & \sum_{k=1}^K c^{(k)} Q^{(k)} \leq \beta. \end{aligned}$$

It can then be solved using any convex solvers.

Robust Optimization and Robust Portfolio Construction with RW divergence

As another application of RW divergence we discussed in Chapter 3, we discuss ways to use RW divergence to construct ambiguity sets for distributionally robust optimization (DRO) problems, in which statistical divergences are of critical importance. We will show that using RW divergences as a divergence function in DRO problems leads to simple forms of asymptotically valid ambiguity sets, and discuss the construction of robust portfolios under mean-variance framework.

Consider the following setting in robust optimization and machine learning:

$$\min_{\theta} \max_{\mathbb{P} \in \mathcal{P}} \mathbb{E}_{\mathbb{P}}[\ell(X, Y; \theta)],$$

which minimizes the loss function ℓ with available features X (and potentially labels Y in the supervised learning setting) over the parameter θ . The data follows probability distribution \mathbb{P} , which is allowed to vary inside an ambiguity set \mathcal{P} . In the data driven setting where iid samples $\{X_i\}_{i=1}^n$ are drawn from an underlying probability distribution \mathbb{P} , we consider ambiguity sets defined as the Relaxed-Wasserstein ball centered at the empirical distribution subject to certain constraints: $\mathcal{P} = \{\mathbb{P} : W_{D_{\phi}}(\mathbb{P}, \mathbb{P}_n) \leq \delta, \mathbb{E}_{\mathbb{P}}[h(X, \theta)] = 0\}$.

Now let X be a random variable in \mathbb{R}^m , with i.i.d copies X_1, \dots, X_n , $\theta \in \mathbb{R}^l$ be the model parameter of interest, and $h(\cdot, \cdot)$ be the optimality condition of the parameter θ to be calibrated. Notice that h also restricts the type of distributions we put in the ambiguity set, for example, if $h(X, \theta) = X^n - \theta$, then we restrict the n th moment of the distribution to be θ . Then one can easily extend Proposition 1 in Blanchet, Y. Kang, and Murthy (2016) to RW divergence, with little modification of the proof:

Proposition 2. *Let $h(\cdot, \theta) : \mathbb{R}^m \times \mathbb{R}^l \rightarrow \mathbb{R}^r$ be Borel measurable and integrable, and $\Omega = \{(u, x) \in \mathbb{R}^m \times \mathbb{R}^m : D_{\phi}(u, x) < \infty\}$ be Borel measurable and non-empty. Further, suppose that 0 lies in the interior of the convex hull of $\{h(u, \theta) : u \in \mathbb{R}^m\}$. Define the Robust Wasserstein Profile (RWP) function as*

$$R_n(\theta) = \inf\{W_{D_{\phi}}(\mathbb{P}, \mathbb{P}_n) : \mathbb{E}_{\mathbb{P}}[h(X, \theta)] = 0\}.$$

Then the following duality result holds,

$$R_n(\theta) = \sup_{\lambda \in \mathbb{R}^r} \left\{ -\frac{1}{n} \sum_{i=1}^n \sup_{u \in \mathbb{R}^m} \{\lambda^T h(u, \theta) - D_{\phi}(u, X_i)\} \right\}. \quad (5.6)$$

From Equation 5.6, we can derive the limiting distributions of $R_n(\theta)$. Two examples will be given later. If the limiting distribution of $R_n(\theta)$ is known, then the $\alpha/2$ and $(1 - \alpha/2)$ quantiles of the limiting distribution would give a $1 - \alpha$ confidence region for $R_n(\theta)$. These quantiles provide an ambiguity set of the smallest size, such that the set contains the true distribution \mathbb{P} underlying the data. Inverting these quantiles would further yield a $1 - \alpha$ confidence region for the θ .

Here we present two examples of special choices of ϕ . For simplicity, let $h(x, \theta) = x - \theta$. Then the constraint $\mathbb{E}_{\mathbb{P}}[h(X, \theta)] = 0$ becomes $\mathbb{E}_{\mathbb{P}}X = \theta$. This is saying that the first moment of the distribution in the ambiguity set should be θ . Choose $c(u, x) = D_{\phi}(u, x)$ for any strictly convex ϕ . Proposition 2 then implies

$$\begin{aligned} R_n(\theta) &= \sup_{\lambda \in \mathbb{R}} \left\{ -\frac{1}{n} \sum_{i=1}^n \sup_{u \in \mathbb{R}} \{ \lambda(u - \theta) - D_{\phi}(u, X_i) \} \right\} \\ &= \sup_{\lambda \in \mathbb{R}} \left\{ \lambda\theta - \frac{1}{n} \sum_{i=1}^n \sup_{u \in \mathbb{R}} \{ \lambda u - \phi(u) + \phi(X_i) + \phi'(X_i)(u - X_i) \} \right\}. \end{aligned}$$

We know

$$\sup_u \{ \lambda u - \phi(u) + \phi'(X_i)u \} = \psi(\lambda + \phi'(X_i)),$$

where ψ is the convex conjugate of ϕ . Then

$$R_n(\theta) = \sup_{\lambda \in \mathbb{R}} \left\{ \lambda\theta - \frac{1}{n} \sum_{i=1}^n \{ \phi(X_i) - \phi'(X_i)X_i + \psi(\lambda + \phi'(X_i)) \} \right\}. \quad (5.7)$$

Example 3. If $\phi(x) = x^2$, which corresponds to the L^2 distance, then Equation (5.7) is reduced to

$$R_n(\theta) = \left(\frac{1}{n} \sum_{i=1}^n (X_i - \theta) \right)^2.$$

Applying the Central Limit Theorem, $R_n(\theta)$ converges to a χ^2 distribution.

Example 4. Take $\phi(x) = x \log x - x$, which corresponds to the KL divergence, then $\phi'(x) = \log x$, $\psi(x) = \psi'(x) = e^x$. The first order condition for λ in Equation (5.7) gives us

$$\theta = \frac{1}{n} \sum_{i=1}^n \psi'(\lambda + \phi'(X_i)).$$

Solve for λ and we get

$$\lambda = \log \theta - \log \left(\frac{1}{n} \sum_{i=1}^n X_i \right).$$

Then

$$R_n(\theta) = \theta \log \theta - \theta - \theta \log \left(\frac{1}{n} \sum_{i=1}^n X_i \right) + \frac{1}{n} \sum_{i=1}^n X_i,$$

which by the Central Limit Theorem and the continuous mapping theorem, it converges to a normal distribution plus the logarithm of a normal distribution. The parameters of the limiting distribution can be easily estimated using data. Moreover, to construct a $1 - \alpha$ confidence region for θ , one only needs to find the $\alpha/2$ and $(1 - \alpha/2)$ quantiles of the limiting distribution.

Now we specialize to the mean-variance portfolio construction problem, which is to construct a portfolio with a required expected return such that the risk, measured by the variance of the portfolio, is minimized. A robust version of this mean variance portfolio problem is to consider all possible distributions for the returns of assets in the ambiguity set, and then to optimize under the worst case scenario.

Mathematically, let $\pi \in \mathbb{R}^d$ be the vector of the weights of d assets in the portfolio, $Var_{\mathbb{P}}(R) \in \mathbb{R}^{d \times d}$ be the variance of the vector of returns R under a probability measure \mathbb{P} , $\mathcal{U}(\mathbb{P}_n) = \{\mathbb{P} : W_{D_\phi}(\mathbb{P}, \mathbb{P}_n) \leq \delta\}$ be the ambiguity set, which is a RW ball centered at the empirical distribution \mathbb{P}_n , and $\mathcal{F}_{\delta, \alpha}(n) = \{\pi : \pi^T \mathbf{1} = 1, \min_{\mathbb{P} \in \mathcal{U}_\delta(\mathbb{P}_n)} [E_{\mathbb{P}}(\pi^T R)] \geq \alpha\}$ be the feasible region of π such that the portfolio has minimum return of α in the worst case. The mean-variance portfolio construction problem with a certain underlying probability distribution \mathbb{P} can then be formulated as:

$$\min_{\pi^T \mathbf{1} = 1, E_{\mathbb{P}}(\pi^T R) \geq \alpha} \pi^T Var_{\mathbb{P}}(R) \pi. \quad (5.8)$$

The distributionally robust version of Equation 5.8 with Relaxed Wasserstein ambiguity set is then the following min-max problem:

$$\min_{\pi \in \mathcal{F}_{\delta, \alpha}(n)} \max_{\mathbb{P} \in \mathcal{U}_\delta(\mathbb{P}_n)} \pi^T Var_{\mathbb{P}}(R) \pi.$$

If $\phi(x) = x^2$, then according to Theorem 1 in Blanchet, Lin Chen, and X. Y. Zhou (2018), the following duality result holds:

$$\min_{\pi \in \mathcal{F}_{\delta, \alpha}(n)} \max_{\mathbb{P} \in \mathcal{U}_\delta(\mathbb{P}_n)} \pi^T Var_{\mathbb{P}}(R) \pi = \min_{\pi \in \mathcal{F}_{\delta, \alpha}(n)} \left(\sqrt{\pi^T Var_{\mathbb{P}_n}(R) \pi} + \sqrt{\delta} \|\pi\|_2 \right)^2. \quad (5.9)$$

That is to say, the distributionally robust optimization problem of minimizing portfolio variance is equivalent to minimizing the standard deviation of the portfolio under the empirical distribution, plus an L^2 penalization term. In other words, to counter the effect of a adversarially-chosen distribution in the ambiguity set (to reduce variance), we should just add an additional regularization to the original problem (Equation 5.8) (adding bias). This is similar to our previous discussion about bias-variance tradeoff in Section 2.1.

5.2 MetaAutoEncoder in Metagenomic Binning

In this section, we will talk about the application of variational autoencoders to the problem of metagenomic binning.

Introduction

In bioinformatics and in the study of metagenomics, shotgun sequencing is used to produce short reads from DNA sequences in a sample from a microbial community, which could

contain thousands species of discovered or unknown microbes, each species carrying its own genome. The short reads are then assembled by connecting overlapping subsequences and thus form longer sequences called contigs. A contig is a short DNA segment. It is continuous and consists of nucleotides. In a DNA sequence, nucleotide comes with 4 different kinds of nucleobases: Adenine (A), Thymin (T), Cytosine (C) and Guanine (G). Their permutations in the contigs encode information into genes.

A typical bacterial genome has a length of around 5 million base pairs. In comparison, human has around 500 million base pairs on one chromosome. However, a modern sequencing machine can read $2 * 300$ basis pairs at an error rate of 0.2%, or $2 * O(10^3)$ basis pairs at an error rate of 14%, which is still short compared with the length of a genome. Therefore, the genomes shall be fragmented into reads, assembled, and then binned into clusters. Metagenomic binning is the process of grouping contigs from multiple organisms into bins. Ideally, each bin should contain readings from just one species. The whole procedure is illustrated in Figure 5.2.

In this section, we discuss the application of deep learning techniques to generate more informative encodings of features extracted from the samples, such that contigs from different species are more separable under the cosine similarity measure, and the performance of the clustering algorithms are improved.

Related Algorithms

Many metagenomic binning algorithms have been developed over the years. Some algorithms rely solely on information of nucleotide composition and abundance. For example, MetaBAT (D. D. Kang et al. 2015) and MetaBAT2 (D. Kang et al. 2019) use probabilistic models and Bayesian statistics to compute similarities between contigs and VAMB (Nissen et al. 2018) uses deep learning (variational autoencoder) to extract low-dimensional representations from high-dimensional features. These algorithms rely on the empirical fact that the distribution of different k-mers of nucleotides are usually characteristic for a certain species in a sample. Here k-mer is defined as a subsequence of length k in DNA (nucleotide) sequences.

Some other algorithms that work on metagenomic datasets are reference-based. For example, SPHINX (Mohammed et al. 2010), CARMA (L. Krause et al. 2008), MGmapper (Petersen et al. 2017) and MetaPhlan2 (Truong et al. 2015). These kind of algorithms usually classify contigs into existing taxonomic groups (sometimes on the levels of species, genus, family, order, class, up to phylum) based on existing reference genome libraries that have already been sequenced and analyzed.

Algorithms such as Chatterji et al. (2008) use the reference genomes in a semi-supervised fashion. That is, their feature vector is GC-contents, k-mer histogram, etc., but the clustering algorithm that follows chooses thresholds that are based on the existing references. Here GC-content refers to the percentage of Guanine (G) and Cytosine (C) in the sequence of DNA.

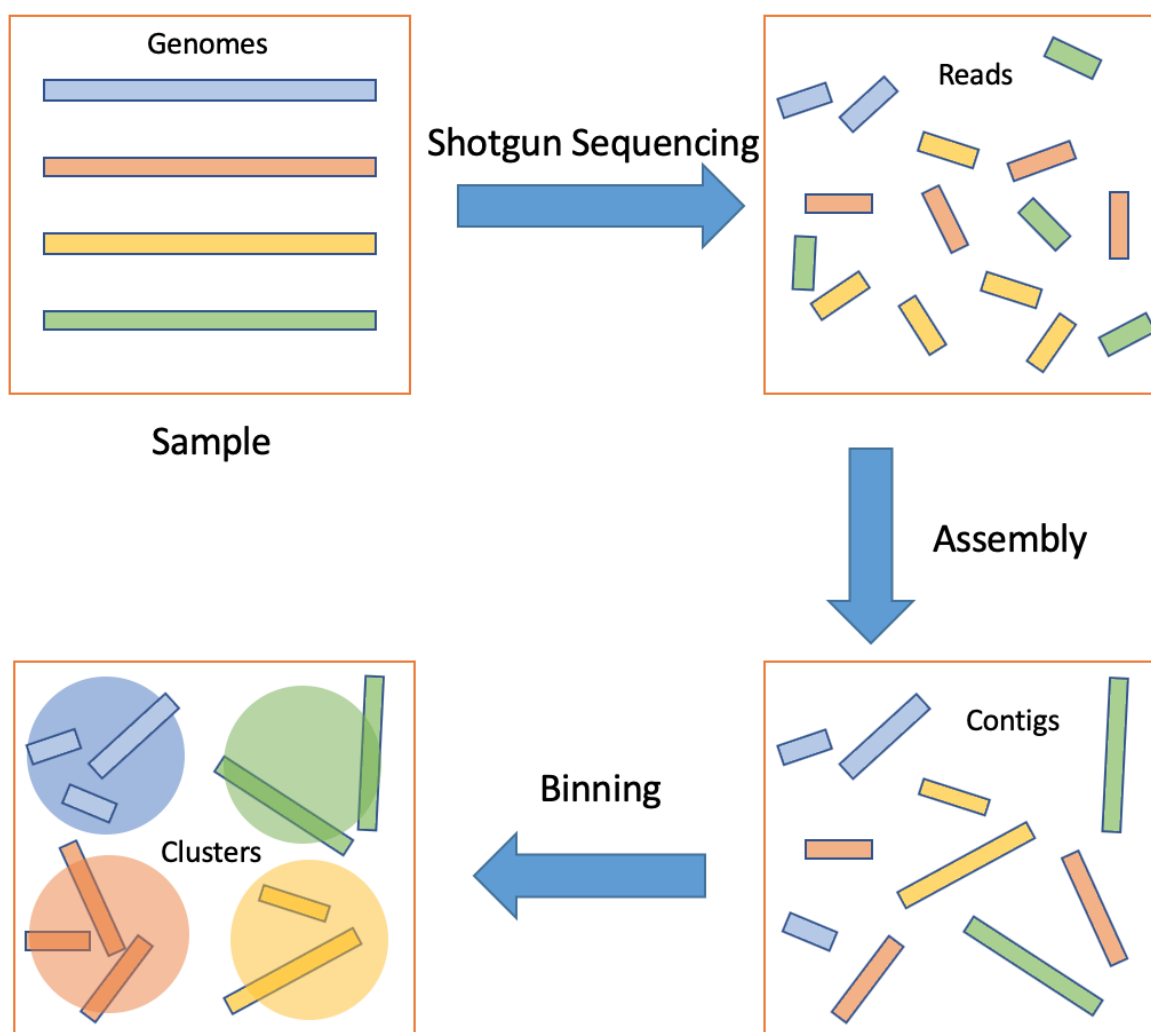


Figure 5.2: Metagenomics

Model Formulations

In this section, we introduce an improved metabinining algorithm called MetaAutoEncoder. The structure of our model is described in Figure 5.3.

The model takes four matrices, **Abundance**, **TNF**, **NT** and **AA**, as inputs. We feed **NT** and **AA** into two separate autoencoders and keep their latent encoding. In the next step, a variational autoencoder encodes the encodings and the original **Abundance** and **TNF** matrices, and we take only the encodings (latent representations). In the next step, we compute the cosine similarity distance between each pair of the encodings. Finally a clustering algorithm is invoked to group the contigs into bins.

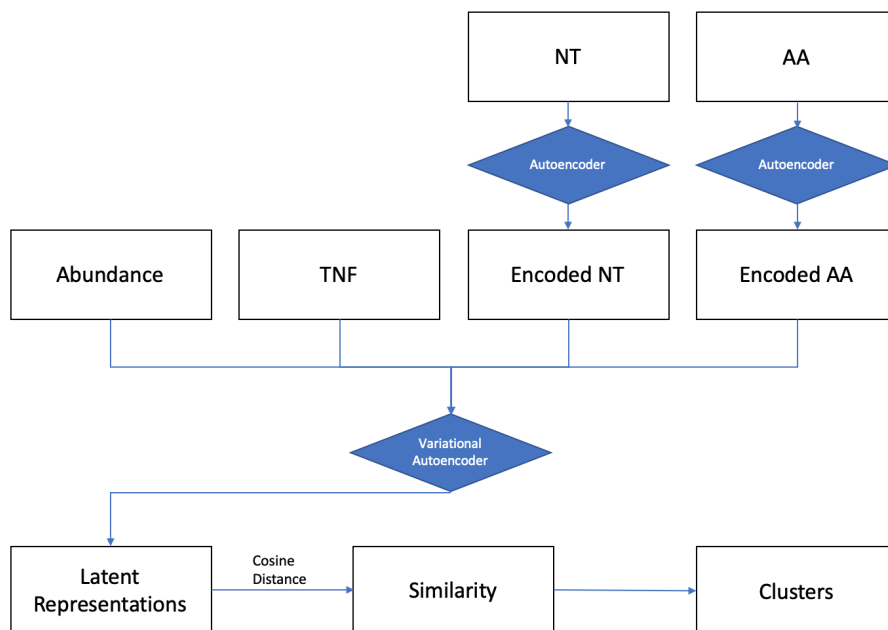


Figure 5.3: Model Structure

Model Inputs

Our model uses the following matrices as inputs:

- Abundance matrix (also known as **Depth**). It describes the abundance of contigs in each sample. After normalization, it can be treated as a distribution among samples.
- Tetra-Nucleotide Frequency matrix (**TNF**). It describes the genome sequence composition. It keeps track of the number of appearances (as in a histogram) of each of the 136 permutations of 4 nucleotides (also known as 4-mers) in every contig.
- NeucleoTide Sketch similarity matrix (**NT**). It is a reference-genome similarity matrix. Each row in **NT** is the similarity of the contig's similarity to a set of known genomes at the nucleotide level. Each entry in **NT** is a real number between 0 and 1.
- Amino Acid Sketch similarity matrix (**AA**). It is a reference-genome similarity matrix. Each row in **AA** is the similarity of the contig's similarity to a set of known genomes at the amino acid level. Each entry in **NT** is a real number between 0 and 1.

Table 5.1 summarizes the descriptions and sizes of the input matrices. In the table, N is the number of contigs, S the number of samples, K_1 the number of reference genomes for **NT**, and K_2 the number of reference genomes for **AA**.

Table 5.1: Input Matrices

Matrix Name	Shape	Type	Description
Label	$(N, 1)$	String	Contains genome IDs
TNF	$(N, 136)$	Dense floats	Contains tetra-nucleotide frequencies
Depth	(N, S)	Dense floats	Contains abundance information
NT	(N, K_1)	Sparse floats	Reference similarity matrix (nucleotide)
AA	(N, K_2)	Sparse floats	Reference similarity matrix (amino acid)

Preprocessing

The TNF matrix and the NT / AA matrices are standardized by each column. That is, we subtract the mean from each column of the matrix and then divide it by the standard deviation. If a column is constant for all contigs, then we just put a uniform distribution on it.

The Abundance matrix is normalized by each row using the L^1 norm. That is, we divide each row of the matrix by the sum of that row so that the sum of the normalized row is equal to 1. If a row is empty for a contig, we put a uniform distribution on it so that each entry is equal to $\frac{1}{N}$.

Network Structure and training

Autoencoder. It is a vanilla autoencoder, characterized by an encoding network and a decoding network. The default structure of the encoding network is [1000, 500, 250, 100]-dimensional dense layers with leaky ReLU as activation functions. The default structure of the decoding network is [250, 500, 1000, K_1 (or K_2)]-dimensional dense layers with leaky ReLU as activation functions. The parameter for the leaky ReLU function is set to 0.2.

The autoencoder is trained on the NT and AA matrices for dimensionality reduction and feature extraction purposes. The encodings are no longer sparse. By dimension reduction we also reduce the number of parameters in the next variational autoencoder network, hence increasing training speed and reducing memory usage.

The loss function of this autoencoder is L^2 reconstruction loss. It is formulated in Equation 5.10,

$$\ell_{\text{ae}} = \sum_{i=1}^N \|\mathbf{x}_{i,\text{true}} - \mathbf{x}_{i,\text{reconstructed}}\|_2^2. \quad (5.10)$$

Here $\mathbf{x}_{i,\text{true}}$ is the i th row of the NT matrix, and $\mathbf{x}_{i,\text{reconstructed}}$ is the output of the autoencoder by feeding in the i th row of the NT matrix.

The network is trained using the Adam optimizer with learning rate = $1e^{-4}$. The default batch size is 200, and the number of epochs is chosen to be 300.

Variational autoencoder. A variational autoencoder is trained on the `Depth`, `TNF`, (encoded) `NT` and (encoded) `AA` matrices to reduce dimensionality and combine features in a non-linear way. The use of the variational autoencoder is to ensure that the encodings are scattered in the latent space as evenly as possible.

The current structure of the variational autoencoder is as follows. By default, on the encoder side first goes through a 325-dimensional dense layer with leaky ReLU activation, followed by a dropout layer with drop rate 0.2 and a batch normalization layer. The parameter of the leaky ReLU function is set to be 0.01.

After this, the output goes through another dense-activation-dropout-batchnorm combination with the same configurations. The reparameterization layer then samples from a Gaussian distribution with mean and standard deviation being the outputs from the previous layer. After this, the decoder has exactly the reverse structure of the encoder.

The loss function of this variational autoencoder is a combination of L^2 reconstruction loss, cross entropy loss and KL divergence penalty. In particular,

$$\ell_{\text{vae}} = \alpha * \ell_{\text{sse}} + \frac{1 - \alpha}{S} * \ell_{\text{ce}} + \frac{1}{n_{\text{latent}}\beta} * \ell_{\text{KL}}. \quad (5.11)$$

The ℓ_{sse} represents the L^2 reconstruction loss for `TNF`, `NT` and `AA`,

$$\begin{aligned} \ell_{\text{sse}} = & (1 - \gamma_1 - \gamma_2) * \frac{1}{136} \sum_{i=1}^N \|\mathbf{x}_{i,\text{true, tnf}} - \mathbf{x}_{i,\text{reconstructed, tnf}}\|_2^2 \\ & + \gamma_1 * \frac{1}{K_1} \sum_{i=1}^N \|\mathbf{x}_{i,\text{true, NT}} - \mathbf{x}_{i,\text{reconstructed, NT}}\|_2^2 \\ & + \gamma_2 * \frac{1}{K_2} \sum_{i=1}^N \|\mathbf{x}_{i,\text{true, AA}} - \mathbf{x}_{i,\text{reconstructed, AA}}\|_2^2. \end{aligned}$$

The ℓ_{ce} represents the cross entropy loss for reconstructing `Depth`. It is defined as

$$\ell_{\text{ce}} = - \sum_{i=1}^N \sum_{j=1}^S x_{ij,\text{true, Depth}} \ln x_{ij,\text{reconstructed, Depth}}.$$

Here $x_{ij,\text{true, Depth}}$ is the entry in the i th row, j th column of the `Depth` matrix. The KL divergence measures the distance between the reparametrized distribution after the encoding step and the standard Gaussian distribution. It is defined as

$$\ell_{\text{KL}} = -\frac{1}{2} * \sum_{i=1}^N (1 + \|\log \sigma_i\|_2^2 - \|\mu_i\|_2^2).$$

Here μ_i and σ_i are the encoded mean and standard deviation of the i th contig.

During the training step, the data are fed in batches of 64. The batchsize doubles when we reach epoch 25, 75, 150 and 300. The optimizer is the Adam optimizer with learning rate being $1e^{-3}$.

Similarity Matrix and Clustering Algorithms

After we obtain the latent representations from the variational autoencoder, the next step is to compute a similarity matrix and run clustering algorithms. The current model chooses to use the same methods proposed in Nissen et al. (2018). For completeness of presentation, we describe the procedures in details in this section.

The similarity metric is chosen to be the cosine similarity. In particular, the similarity between the representations of two contigs \mathbf{x} and \mathbf{y} in \mathbb{R}^m is defined as

$$\text{similarity} = \frac{\mathbf{x} \cdot \mathbf{y}}{\|\mathbf{x}\|_2 \|\mathbf{y}\|_2}$$

The similarity is always between 0 and 1. If either $\|\mathbf{x}\|_2$ or $\|\mathbf{y}\|_2$ is 0, we define the similarity to be 0.

The clustering algorithm is a modified version of the k -medoid algorithm, which is described in Algorithm 6.

Algorithm 6 Clustering Algorithm (iterative medoid clustering)

Require: *contigs*, list of contigs, *threshold*, threshold of distance, N_{samples} , number of samples

```

1: seed  $\leftarrow$  first contig of contigs
2: cluster  $\leftarrow$  set of all contigs within threshold of seed
3:  $D(\text{cluster}) \leftarrow$  mean distance of seed to each other member of cluster
4: for trialseed of  $N_{\text{samples}}$  randomly sampled contigs from cluster do
5:   trial  $\leftarrow$  set of all contigs within threshold of trialseed
6:    $D(\text{trial}) \leftarrow$  mean distance of trialseed to each other member of trial
7:   if  $D(\text{trial}) < D(\text{cluster})$  then
8:     seed  $\leftarrow$  trialseed
9:   go to 2
10: end if
11: end for
12: output cluster
13: contigs  $\leftarrow$  contigs \ cluster
14: if contigs  $\neq \emptyset$  then
15:   go to 1
16: end if

```

In the clustering algorithm, the threshold is determined using the following procedure, described in Nissen et al. (2018). First, 2500 contigs are sampled from the dataset. For each contig, its cosine distances from all other contigs are calculated. We then calculate and draw a histogram for the distances.

Two groups are said to be well-separated if two peaks can be spotted in the histogram and the density falls below 0.025 between the two peaks. The threshold is then defined as the distance between the two points where the density is equal to 0.025.

Two groups are said to be poorly-separated if the density between peaks falls below 80% of the density at the larger peak. The threshold is then defined as the distance between the two points where the density is equal to $0.8 * \text{peak density}$.

For other patterns of the histogram, no threshold will be given because the groups cannot be easily separated using the cosine similarity.

Finally, the threshold used in the clustering algorithm is set to be the median of thresholds obtained in the previous step.

Performance Evaluation

Experiment setting

The model is tested on two independent synthetic datasets: the high-definition MetaHIT dataset and the high-definition ActinoMock dataset. The reason we use synthetic datasets is to ensure that the ground truth is clear: the genomes of the bacteria grown in the samples have been accurately recorded in the library, hence the label of each contig is known.

MetaHIT is a large synthetic dataset in which each contig is simulated from a sequenced reference genome. It contains 169864 labeled contigs from 256 genomes and 264 samples.

ActinoMock (high-D) is another synthetic dataset, independent from MetaHIT. It is smaller: it only contains 1188 labeled contigs from 12 genomes and 2 samples.

In our experiments, we keep all contigs regardless of their lengths.

Clustering Performance

In clustering analysis, a requirement of precision at 0.95 means that 95% of the contigs a certain bin contains should come from the same genome. On the other hand, a requirement of recall at 0.9 means that 90% of the contigs from a certain genome should be identified and recovered in a single bin. Recall and precision together can tell us how the clusters recover the original genomes from the dataset. Figure 5.4 shows the performance of our algorithm (MetaAE) versus VAMB on the ActinoMock dataset under different precision and recall requirements. Figure 5.5 shows the performance of our algorithm (MetaAE) versus VAMB on the MetaHIT dataset under different precision and recall requirements. The MetaAE algorithm improves both the quality of bins as well as the number of bins recovered compared to VAMB, in both precision requirement settings.

Discussions

In this section, we try to answer the following questions:

1. What is the reconstruction quality for the autoencoders?
2. Does the NT/ AA matrix add additional information? If so, how should we include this information to the original framework?

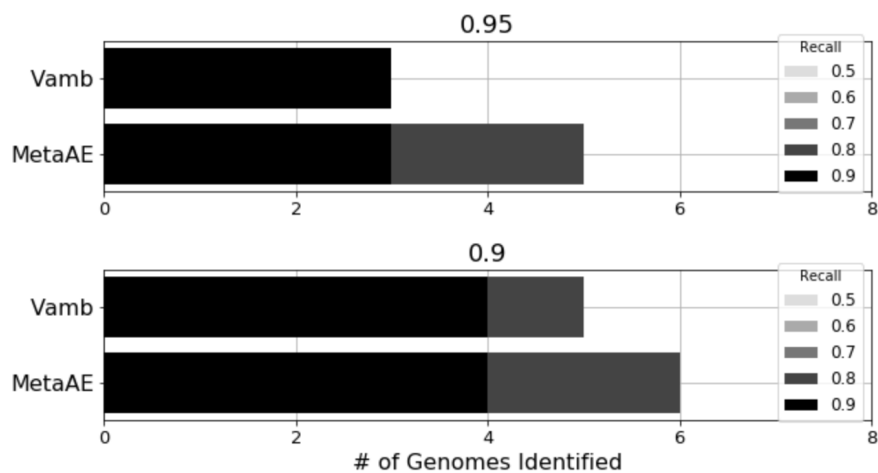


Figure 5.4: Clustering results of MetaAE vs. VAMB on the ActinoMock dataset

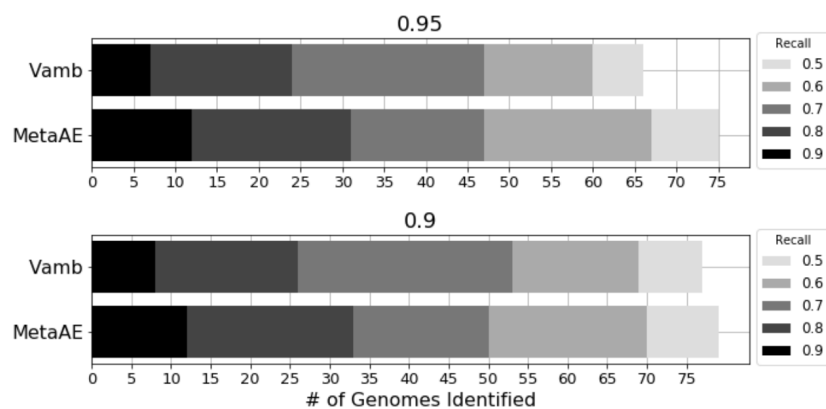


Figure 5.5: Clustering results of MetaAE vs. VAMB on the MetaHIT dataset

3. Should we use autoencoders, or SVD / PCA?
4. How do we deal with uncertainty in the NT / AA similarity measurements?
5. How will this method work if we have only partial information available, i.e., what if the abundance matrix is not available?

What is the reconstruction quality for the autoencoders?

The NT and AA matrix both have high dimensions. They are also very sparse. Table 5.2 describes the row-wise and column-wise sparsity in the MetaHIT dataset. In this case, the dot product of two rows would mostly result in 0. Hence, using cosine distance to measure the similarity between rows does not make much sense.

Table 5.2: Sparsity in the MetaHIT dataset

Percentile	Row	Column	Percentile	Row	Column
0 (min)	0	1			
10	1	2	60	12	154
20	2	5	70	25	242
30	3	11	80	34	1009
40	5	25	90	81	7011
50	8	62	100 (max)	4458	13973

In the latent space, however, the low-dimensional representation is dense. Besides, the NT / AA matrix is usually huge in size. Using autoencoder avoids reading the whole matrix, but rather reads a batch of rows at a time. In this experiment, we do not enforce a penalty such that the reconstructed rows shall be sparse. The reconstruction error for the MetaHIT dataset is presented in Table 5.3. The reconstruction error (RE) is measured using sum of squared errors. For most of the rows, the reconstruction error is below 20%.

Table 5.3: Reconstruction Errors of the MetaHIT dataset using Autoencoders

Percentile	SS per row	RE	RE in percentage
0	0	86	0.03
10	10000	316	0.20
20	20000	460	0.52
30	29946	620	0.82
40	49296	865	1.44
50	75506	1203	2.43
60	116367	2087	4.04
70	234320	4705	6.21
80	325090	11014	9.16
90	774354	35636	18.01
100	40487081	1901078	inf

How do the matrices separate the clusters?

We randomly sample 15 genomes from the MetaHIT dataset and collect the contigs. For each pair of the contigs, we count the number of non-zero entries (columns) that are shared by them in the NT matrix. Figure 5.6 compares the normalized histogram of the counts of intra-genome pairs and inter-genome pairs. It can be seen that although the intra-genome pairs tend to have more non-zero columns in common, over half of the time the number of shared non-zero entries is still very low and hence this could not be reliably used to determine whether two contigs come from the same genome.

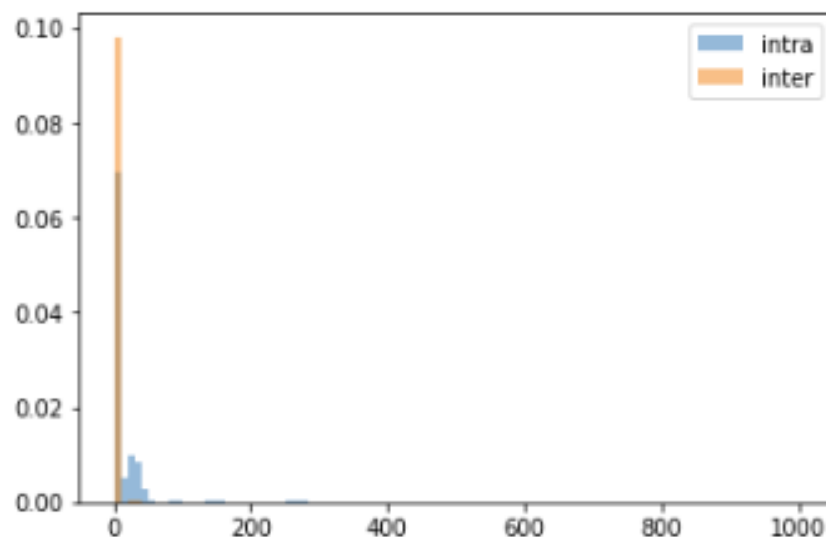


Figure 5.6: Histogram of numbers of pairwise nonzero elements of inter-genome and intra-genome contigs in the NT matrix on the MetaHIT dataset

However, if we compute the cosine similarity of every pair of contigs using rows from the NT matrix, the distributions become extremely separable, as observed in Figure 5.7. As a comparison, Figure 5.8 shows how the similarity between TNF separates the genomes.

To make this more concrete, are NT / AA matrices informative? The final goal of metagenome binning is to produce bins that contain similar contigs. In the previous sections, we choose the cosine distance to be the distance metric between contigs, which is $(1 - \text{cosine similarity})$. We then feed the pairwise distance / similarity matrix to the clustering algorithm 6 and get the clusters. Indeed, the number of genomes (species) we can recover is the ultimate criterion of the whole process. However, the clustering performance shown in the above sections relies heavily on the clustering algorithm used to generate clusters. We hence would like to find a simple way to tell how informative the features that our encoded features are in the space of cosine distance. To do that, we propose to use the following performance metric.

Metric. First, we randomly sample k genomes from the dataset without replacement and collect a set of contigs. For each pair of the contigs in the set, their cosine similarity is computed. Then a decision tree model with maximum depth of 3 is trained to maximize the accuracy in predicting whether each pair of contigs belong to the same genome, using the similarities as a 1-dimensional feature. Because the feature is only 1-dimensional, the model is essentially choosing the best threshold to split the intra-cluster (genome) similarities and the inter-cluster (genome) similarities. We then sample another independent testing set, run the decision tree algorithm on it, and output the confusion matrix.

Table 5.4 shows the confusion matrix that uses NT to compute similarities between each

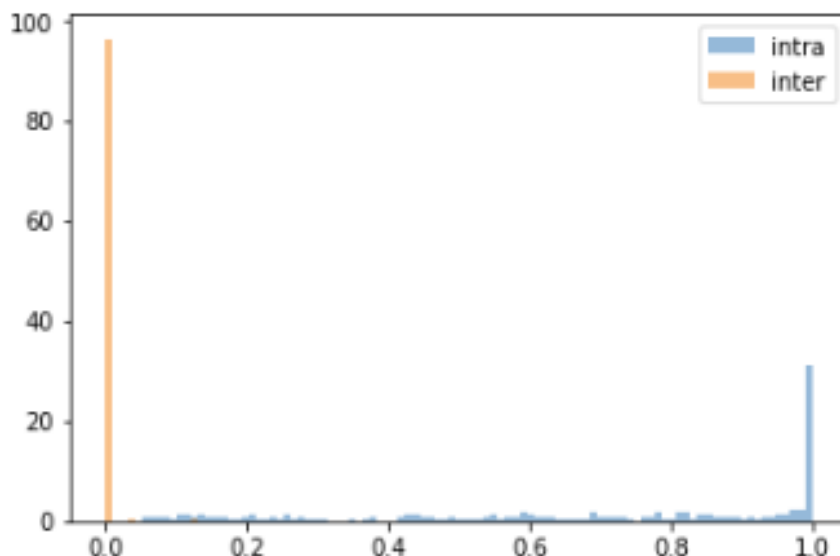


Figure 5.7: Histogram of similarities of inter-genome and intra-genome contigs in the NT matrix on the MetaHIT dataset

pair of contigs. Positive in the matrix means that the contigs are from the same genome, whereas negative means they are from different genomes. Table 5.5 shows the confusion matrix that uses the encoded NT to compute similarities. Notice here NT has 21288 columns whereas the encoded version only has 40 dimensions. In view of the fact that they have similar performances, we could conclude that the encoding is effective. The autoencoder structure here increases training speed by reducing the number of trainable parameters. In addition, the encoded NT can be used alone later for hyperparameter tuning or other tasks.

Table 5.4: Confusion matrix on MetaHIT using NT

		Predicted Outcome		Total
		Positive	Negative	
Actual Outcome	Positive	259749	2751	262500
	Negative	5329	13046	18375
Total		265078	15797	280875

Should we use autoencoders or SVD / PCA for dimension reduction?

To speed up training and to gain better latent representations, we need to reduce the dimensionality of the NT / AA matrices. SVD is an effective and common method for dimension reduction. We apply SVD to both datasets. In terms of dimension reduction, SVD

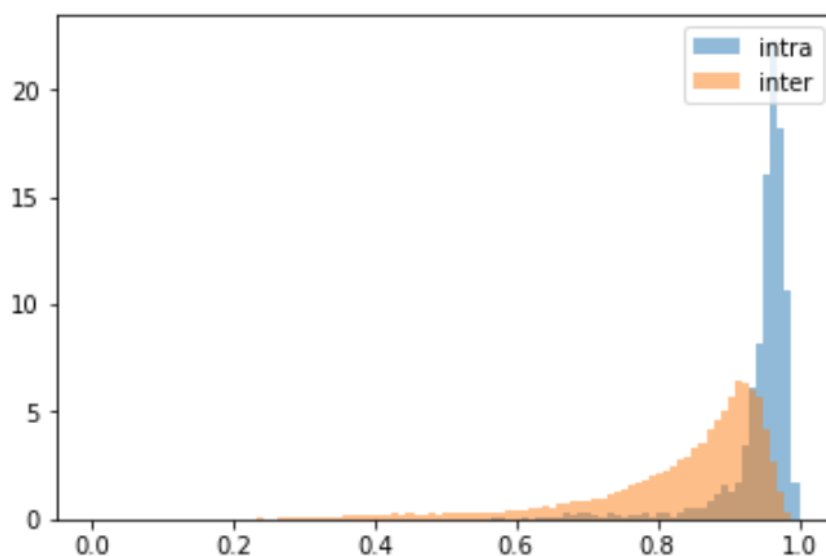


Figure 5.8: Histogram of similarities of inter-genome and intra-genome contigs in the TNF matrix on the MetaHIT dataset

Table 5.5: Confusion matrix on MetaHIT using encoded NT

		Predicted Outcome		Total
		Positive	Negative	
Actual Outcome	Positive	246234	16266	262500
	Negative	5059	13316	18375
Total		251293	29582	280875

works well. On the ActinoMock dataset, by taking the first 100 principle components, we can recover 93.10% of the total variation.

Figure 5.9 compares the clustering results of MetaAE with SVD and VAMB on the MetaHIT dataset. We can see that the inclusion of SVD slightly improves the performance at the 0.95 precision level, but reduces the number of genomes recovered at the 0.9 precision level. This could be due to the fact that SVD fails to describe the non-linear dependencies for the reference genomes, while the autoencoders do.

How do we deal with uncertainty in the NT matrix?

The entries in the NT matrix are actually not homoskedastic. In fact, the lower the similarity is, the more uncertain the similarity becomes. For example, a similarity of 0.95 between a contig and a reference genome is more certain than a similarity of 0.80. The sparsity of the NT matrix is also caused by the fact that a similarity below 0.75 is so unreliable that it is truncated to 0.

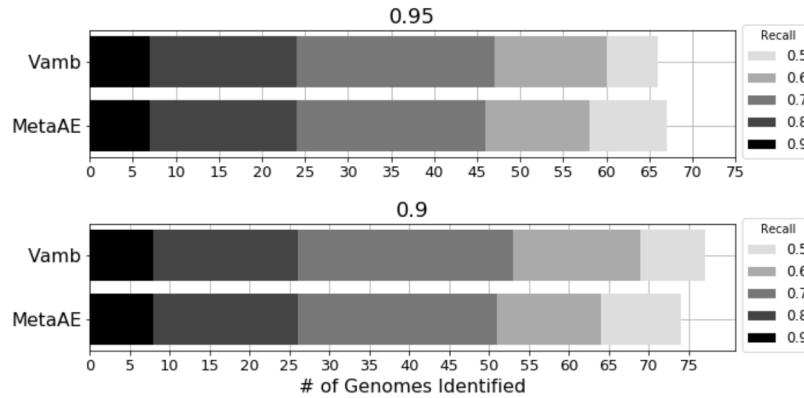


Figure 5.9: Clustering results of MetaAE (with SVD) vs. VAMB on the MetaHIT dataset

This uncertainty from the entries creates two problems: a similarity of 0.75 does not contain more information than a similarity of 0; a similarity of 0.95 should carry more weight in our model than a similarity of 0.80. To solve these two problems, we propose to apply a transformation before autoencoding the NT matrix. In particular, for every entry x_{ij} in the matrix,

$$x'_{ij} = \begin{cases} (x_{ij} - \delta)^2 & \text{if } x \geq \delta, \\ 0 & \text{otherwise.} \end{cases}$$

Here δ is a parameter for thresholding. The new entry will be shifted towards 0 if it is above the threshold. The autoencoder will also make sure that the reconstruction is more accurate when the entry is close to 1.

Figure 5.10 shows the clustering result on the MetaHIT dataset when we set $\delta = 0.75$. It further improves the performance of the MetaAE algorithm.

What if we only have a single sample?

If only a single sample is available, then the **depth** matrix is meaningless and contains no information because every row of it will be normalized to 1. In this case, if we just remove the cross entropy loss from the loss function and rerun the variational autoencoder, the performance is poor: the latent representation can only help identify fewer than 5 genomes on the MetaHIT dataset and fewer than 2 genomes on the ActinoMock dataset.

The **TNF** matrix is noisy and not informative itself. Figure 5.11 compares the performance of **TNF** vs. encoded **TNF** on the MetaHIT dataset. We can see that the overlap of the two distributions is so large that using the similarity based solely on **TNF** does not fully separate intra-genome contigs from inter-genome contigs.

On the other hand, if we use the vanilla autoencoder, the **NT** matrix alone will yield very good performance on the MetaHIT dataset, while combining **TNF** with **NT** does not work well.

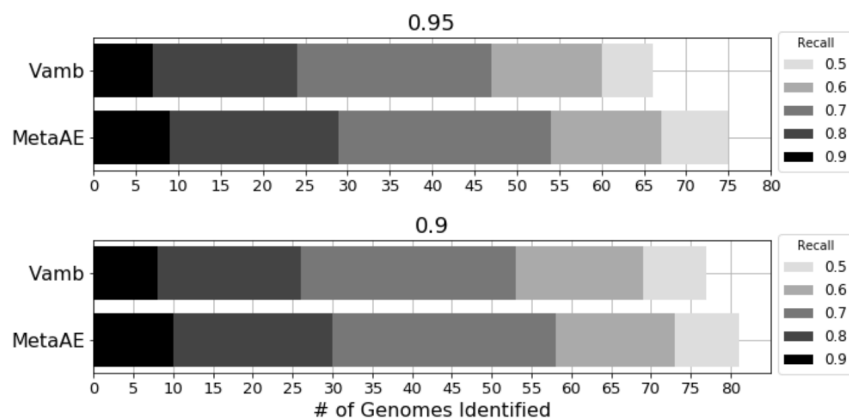


Figure 5.10: Clustering results of MetaAE (with quadratic transform) vs. VAMB on the MetaHIT dataset

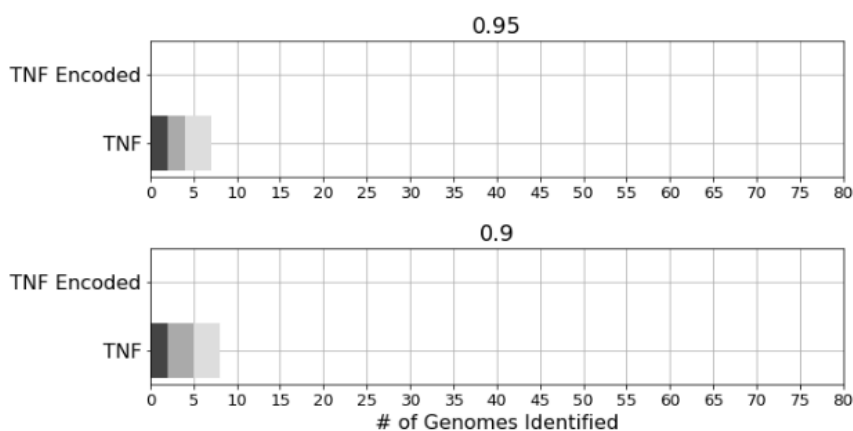


Figure 5.11: Clustering results of TNF vs. encoded TNF on the MetaHIT dataset

Figure 5.12 shows the comparison of performance between using encoded NT and encoded TNF and NT (weight of NT in the reconstruction loss is 0.95) on the MetaHIT dataset.

However, the high performance of the NT matrix is not stable across datasets. On the ActinoMock dataset, for example, NT will not help recover more than 2 genomes, as shown in Figure 5.13.

These results suggest that the most stable features to use in binning are from **Depth** and **NT**. The algorithm becomes less effective if we are missing one of them. Moreover, the variational structure of the **MetaAE** algorithm helps better separate different clusters.

How is the AA matrix compared with NT?

Compared with NT, using the AA matrix to separate clusters has higher specificity, but lower sensitivity. The reasons are mainly two-fold. First, inside the same family of species, multiple

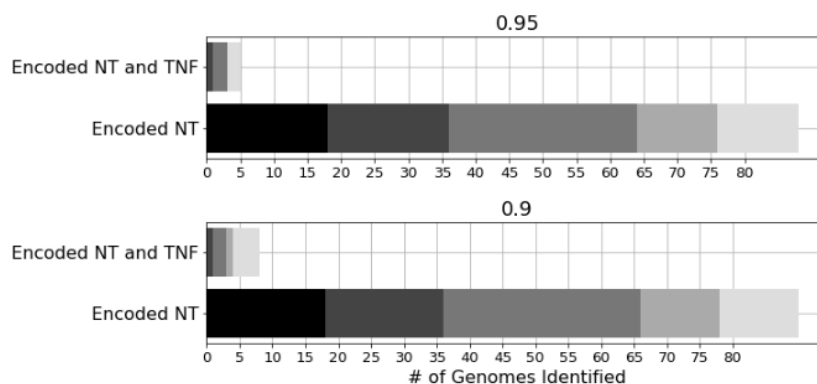


Figure 5.12: Clustering results of encoded NT vs. encoded NT and TNF on the MetaHIT dataset

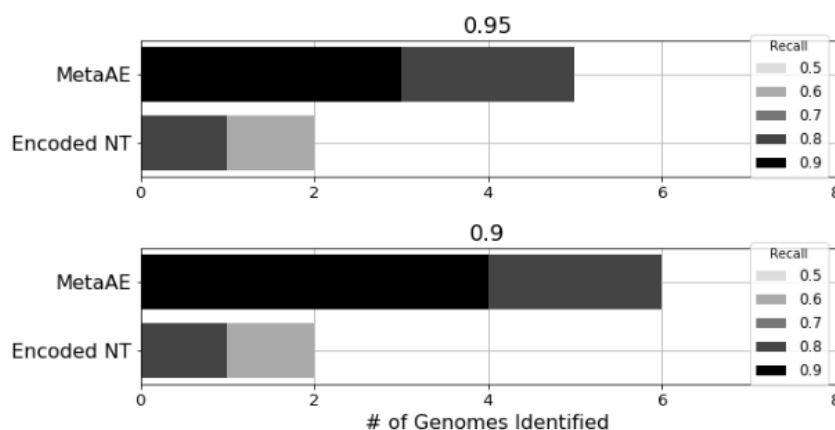


Figure 5.13: Clustering results of encoded NT vs. MetaAE on the ActinoMock dataset

nucleotides could translate to the same amino acids, so similarity measured using AA is usually more ambiguous. Second, with different families of species, the same nucleotide could encode different AA, so similarity measured using NT is more significant. Compared to NT, Figure 5.14 has the peak of inter-species similarity at 0 is lower. So, usually we would prefer to use NT to AA. However, as is going to be shown in the next section, AA is preferred when we have less confidence what families these species belong to.

Existing problems and potential solutions

In this section, we discuss some of the limitations of this algorithm.

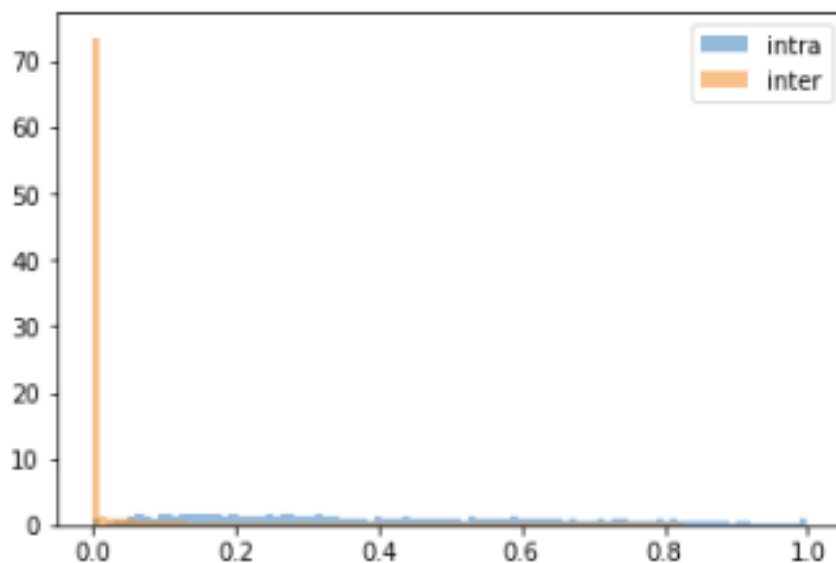


Figure 5.14: Histogram of similarities of inter-genome and intra-genome contigs in the AA matrix on the MetaHIT dataset

The effect of the number of known genomes

First, since the NT matrix measures the similarity between contigs and genomes in a reference library, the number of genomes that already exist in the reference library matters. For illustration, we remove all genomes that appear in the MetaHIT dataset from the reference library (they have perfect correlation with the contigs), then the total number of columns in NT is reduced from 21288 to 21028. The resulting matrix is called NT100.

Figure 5.15 shows the distributions of similarity of intra-genome contigs vs. inter-genome contigs. It can be observed that compared to Figure 5.7 we lose some separability, but the NT100 matrix is still very good at predicting whether two contigs are from the same genome.

If we further reduce the number of reference genomes in the library, we would result in NT99 where reference genomes that are similar to the genomes in the MetaHIT dataset are also removed from the library. Repeating the procedure above, we would get Figure 5.16. Even with this, the prediction power is relatively strong. These results show that the NT matrices are robust in the size of the reference library, which implies that they could be useful in clustering undiscovered genomes.

However, the existing clustering algorithm (Algorithm 6) does not fully exploit their power. Figure 5.17 compares the clustering results using only encoded NT100 or encoded NT99 matrices as features (single sample, no `depth`, or TNF information).

Amino acids as features, on the other hand, are less sensitive to the size of the reference library. Figures 5.18 and 5.19 show that as the reference size shrinks, the similarity matrix still separates intra- and inter-species contigs pretty well. The peak of intra-species similarity

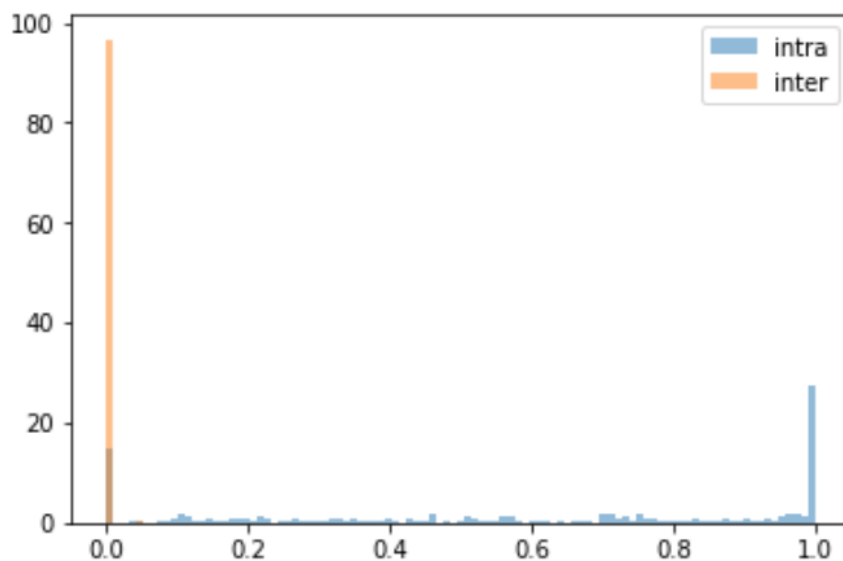


Figure 5.15: Histogram of similarities of inter-genome and intra-genome contigs in the NT100 matrix on the MetaHIT dataset

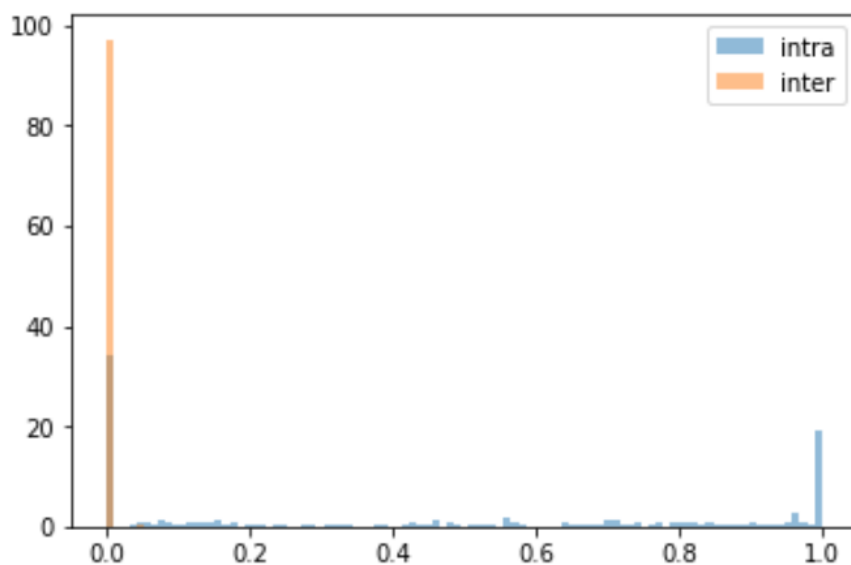


Figure 5.16: Histogram of similarities of inter-genome and intra-genome contigs in the NT99 matrix on the MetaHIT dataset

at 0 is much lower compared to what we have for NT matrices.

A future direction would be to design a metric that measures the coverage of the reference genomes over the sampled contigs. The weights in our Variational Autoencoders should then

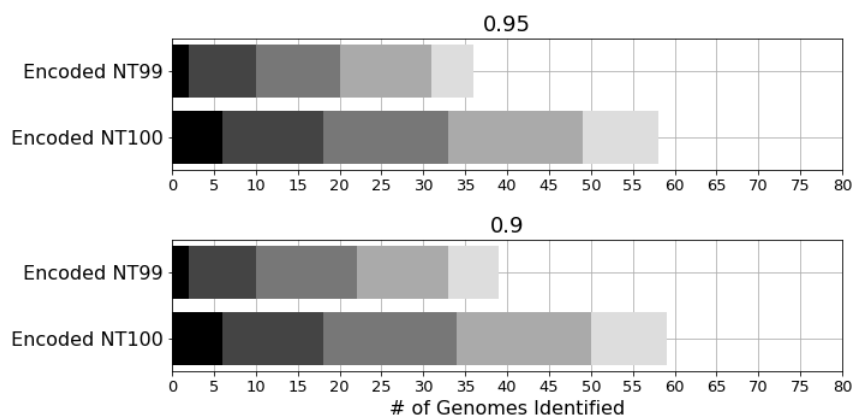


Figure 5.17: Clustering results using only encoded NT99 and encoded NT100 matrix on the MetaHIT dataset

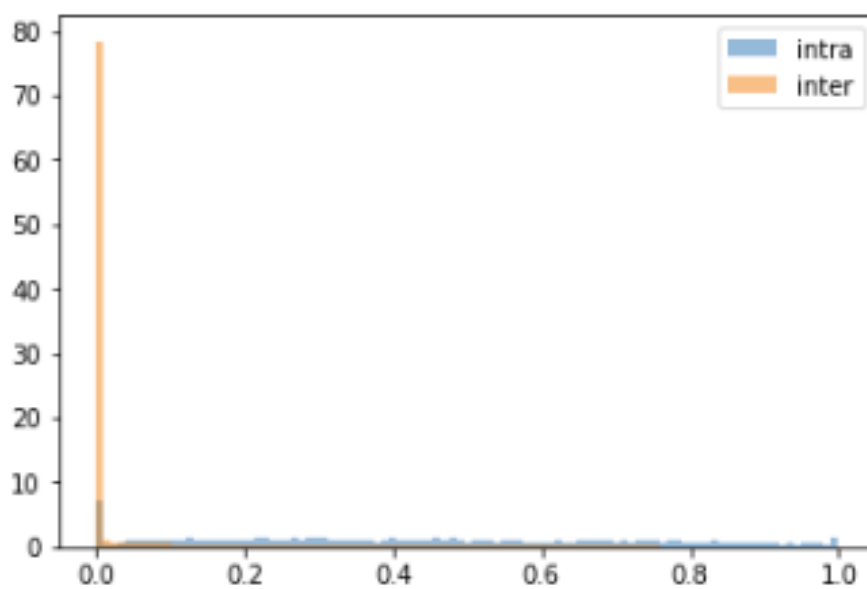


Figure 5.18: Histogram of similarities of inter-genome and intra-genome contigs in the AA100 matrix on the MetaHIT dataset

be adjusted accordingly: the better the coverage, the more weights we should put on the NT matrix.

Conclusion

In this section, we show that variational autoencoders are effective at greatly improving the performance of existing metagenomic binners by incorporating categorical information from

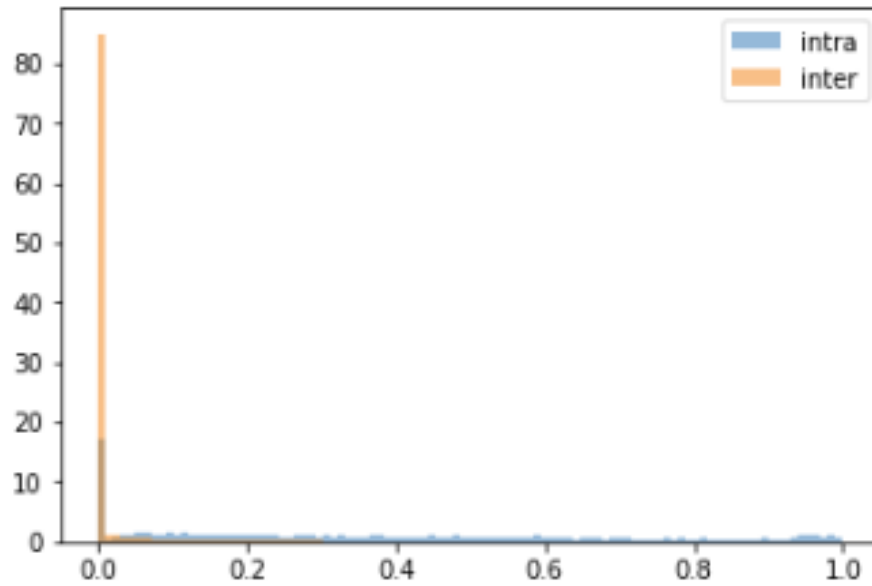


Figure 5.19: Histogram of similarities of inter-genome and intra-genome contigs in the AA99 matrix on the MetaHIT dataset

the reference database. The nonlinear dimension reduction ensures that the generative model is expressive enough to capture the underlying inter- and intra- genomic similarity in the cosine space.

5.3 Simulation of Credit Default Swap Index Transaction Data

From previous chapters, we have seen that GANs are a type of generative model that learns high-dimensional continuous distributions (most often image data). In the last section of this chapter, we present an example showing that GANs have great potential in dealing with financial data as well. In finance, the Credit Default Swap (CDS) index trade data are rare to find and complex in structure. In this section, our goal is to build a simulator that can artificially generate more CDS index trade data that are similar to existing trading records. We train a special type of Generative Adversarial Network, the Sequential GAN (SeqGAN), to achieve this.

After constructing the synthetic dataset, we then explore various ways and metrics to validate the effectiveness of the synthetic dataset. For example, we examine the marginal distributions and the correlations among variables. We also test the relative effectiveness of popular machine learning algorithms on both the original and the synthetic dataset.¹

Introduction

Credit Default Swaps are a type of financial credit derivative designed for buyers to transfer some of the risk exposure on the underlying product, usually corporate bonds, mortgage-backed securities (MBS), or municipal bonds, to the sellers. In the most basic case, the buyers pay the sellers a predetermined price, or spread, on a regular basis. Whenever a special credit event that is specialized in the contract happens (for example, when the underlying bond defaults), the buyer receives payment from the sellers. In essence, the sellers are protecting the buyers from credit risks by issuing insurance.

CDSs are usually highly customized and traded not on exchanges, but over-the-counter. A CDS Index, on the other hand, is a basket of CDSs, which is standardized and traded on the exchange. There are two major types of corporate CDS indices: CDX, which contains companies from North America and emerging markets, and iTraxx. In this section, we will focus on the CDXs. Multiple types of CDXs are available on the market with focus on different risk levels, for example, High Yield (HY) and Investment Grade (IG). The composition of these CDXs are recalibrated and changed regularly (usually every six months). Their prices generally reflect the credit risk evaluated by the market.

Related Works

GANs for data augmentation. Generative neural networks (GANs) are typically used for generating images that are similar to existing ones in the dataset. Since it learns the underlying distribution of observations in a nonparametric way, people have been exploring

¹The methodology used in this section was also used in one of the FinTech projects of the Master of Engineering program at UC Berkeley in Spring 2019.

the uses of GANs in data augmentation and the datasets are not restricted to image data. Data Augmenting GAN (DAGAN) (Antoniou, Storkey, and Edwards 2017) show a new structure of GAN that can improve the accuracy of classifiers by generating additional samples in the low-data regime. Bowles et al. (2018) augment the performance of brain segmentation by introducing a Progressive Growing of GANs (PGGAN) network. Frid-Adar et al. (2018) use a convolutional neural network (CNN) to output synthetic computed tomography (CT) image data in order to improve liver lesion classification. Refer to Mariani et al. (2018), Shin et al. (2018), Nie et al. (2017), and X. Zhu et al. (2017) for more applications.

Simulating CDS. Past efforts in the literature to simulate CDS prices have considered probabilistic techniques. Most of them use credit risk models to simulate default events and generate spreads or prices. To name a few, see Joro, Niu, and Na (2004), Hull and White (2000), Hull and White (2004), and Sethi, Thakkar, and Jamal (2018).

Deep learning in Finance. The use of deep learning in finance, on the other hand, is a relatively new field. Horvath, Muguruza, and Tomas (2019) use deep learning to calibrate stochastic models for volatility surfaces, for example, the rough Bergomi model, the Heston model, and the Bergomi model. The inputs of the network are parameters in the stochastic models; that is, the coefficients of the Brownian motion and the drift terms. The outputs are the volatility at different maturity and strikes. Their method is two-step: 1. Simulate by Monte-Carlo, using the stochastic models and known parameters, thousands of Brownian motion paths as the training set. Then the neural network is optimized by minimizing the mean square error at each grid point of the volatility surface. 2. Simulate another test set. Then try to recover the model parameters using the trained neural network in step 1 (learn the inputs of the networks). In summary, their neural network approximates the mapping between model parameters and the paths of Brownian motions.

Luyang Chen, Pelger, and J. Zhu (2019) adapt the conditional no-arbitrage moment condition to a min-max optimization problem with adversarial loss. The Stochastic Discount Factor weights try to minimize the loss, while the conditioning functions (of macroeconomic and firm-specific characteristics) try to maximize it. The SDF and the conditioning functions can each be approximated by a neural network with LSTM units. This is similar to the GAN structure (two networks with adversarial loss), hence they apply the iterative training procedure from GAN to optimize everything. By definition, the SDF weights are the weights of the conditional mean-variance efficient portfolio, which maximizes the Sharpe ratio.

Feng, He, and Polson (2018) train a neural network, which applies a regression-like model that uses exposures to macroeconomic and firm-wise characteristics in a non-linear fashion, to predict excess asset returns. Gu, B. Kelly, and Xiu (2018) compare deep neural nets with other machine learning models on the performance of measuring asset risk premia. Messmer (2017) also use neural networks to predict asset returns, while Bianchi, Büchner, and Tamoni (2019) predict bond returns. Gu, B. T. Kelly, and Xiu (2019) use autoencoder plus the no-arbitrage assumption to look for nonlinear latent factors that drive asset prices.

Dataset

The dataset is from the public Swap Data Repository (SDR), which is hosted by the Depository Trust & Clearing Corporation (DTCC). According to the U.S. Commodity Futures Trading Commission, the swap dealers are required by the Dodd-Frank Act to report all CDS index (CDX) transactions to registered SDRs. The real-time reporting contains swap transaction and pricing data, available on a daily basis.

We downloaded the credit swap index message data from 01-01-2014 to 02-28-2019, a total of 1885 days, from DTCC's website, among which 30 days of files are missing. After combining the files, the dataset contains a total of 1103307 records with 44 features. Some of the features are categorical variables. For example, the `taxonomy` variable indicates whether the CDX is high-yield (high credit risk) or investment-grade, the `action` variable, like what we usually observe in a limit order book, indicates whether this record is for posting new orders to the market (new), correcting an existing order on the market (correct), or cancel an outstanding order (cancel).

Some variables are continuous in nature. For example, the `price` variable is the price at which the order is posted or traded. However, the price is truncated if it is above certain thresholds that depend on each individual record.

The features are also highly unbalanced. For example, for the `action` feature among all records, 904440 (82%) are new orders, 124583 (11%) are order cancellations, and the rest (less than 7%) are corrections of existing orders. The large fraction of the orders being new is very different from what we usually observe on the equity market, where most orders are cancellations.

The quality of records is relatively low. Ten out of forty-four features are completely unpopulated. Other features contain NaN values, or incorrect values that do not match the descriptions of that column. Preprocessing is hence an essential step in this example.

Preprocessing

We perform a minimum amount of manipulation on the records to ensure that even without much guidance or domain knowledge, the neural network would still be able to capture the essence of the structure of our dataset. The preprocessing therefore focuses on removing redundant or uninformative rows and columns.

For this example, instead of including all 44 variables, of which most are irrelevant or of low data quality, we decide to use 7 of the most relevant features: `action`, `cleared`, `indication_of_collateralization`, `taxonomy`, `price_forming`, `price` and `notional_amount`. The following list summarizes their possible values:

- `action`: new, correct, cancel.
- `taxonomy`: investment grade (IG), high yield (HY).
- `price_forming`: trade, novation, partial termination, amendment.

- `cleared`: cleared, uncleared.
- `indication_of_collateralization`: collateralized, not collateralized.
- `price`: continuous.
- `notional_amount`: continuous.

We proceed through the following steps to preprocess the dataset and prepare for training:

1. Select only orders whose `notional_currency` is U.S. dollars. This reduces the number of rows from 1103307 to 823555.
2. Select only orders whose `taxonomy` is a high-yield or investment-grade credit default swap index (CDX). This reduces the number of orders to 300112.
3. Select only orders with a maturity of around 5 years (4.5 to 5.5 years). The duration is calculated as the difference of `end_date` and `effective_date`. Most of the orders have a maturity in this range. The number of rows is now 285134.
4. To avoid duplication of orders, we drop those actions that create orders which are modified later and those actions that cancel an existing order in our dataset. This way we only consider the most-updated orders. We end up with 245104 orders.
5. We only keep orders whose `price_notation_type` is in basis points or in percentage, leaving us with 158632 orders.
6. We keep orders whose `price` is not 0. This reduces the number of orders to 158131.
7. To deal with the inconsistency in price notations, we normalize the price notations in the following way:
 - For prices quoted in percentage, some of the orders are actually quoted in basis points, which are easily detected if the prices are in the range of [1000, 100000]. We convert them by dividing the price by 10000.
 - For prices quoted in basis points, we convert them to percentage by dividing the price by 100.
8. In the current version of the model, we drop all rows that have an NaN in them. This reduces the number of rows to 30857.
9. To transform the continuous features to discrete ones, we construct bins for the `prices` and the `notional_amounts`.
10. We combine each 30 contiguous records into a sequence. We then add a token `<GO>` to the head of each sequence and append a token `<END>` to the end of each sequence.

Model Setup

In this example, we are dealing with a mixture of continuous and discrete type data; that is, some features are discrete (categorical) and some features are continuous (positive real numbers). In addition, the data are actually a time series: contiguous samples are often posted in a clustered fashion, and therefore their features and prices are correlated. We apply Sequential GAN (SeqGAN) (Yu et al. 2017) to address these two problems through policy gradients. The following section describes how SeqGAN works with a discrete and sequential underlying distribution.

SeqGAN

In SeqGAN, the objective is to train a generative neural network G_θ parameterized by θ to generate a random sequence $Y_{1:T} = (y_1, \dots, y_t, \dots, y_T)$, where $y_t \in \mathcal{Y}$ and \mathcal{Y} is our action space, also called vocabulary in this setting. At the same time, a discriminator D_ϕ , parameterized by ϕ , is trained so that the output on the real data $D_\phi(Y_{1:T}^n)$ is close to 1 and the output on the generated data $D_\phi(Y_{1:T})$ is close to 0. The objective function on the discriminator side is hence similar to a vanilla GAN,

$$\min_{\phi} -\mathbb{E}_{Y \sim p_{\text{data}}}[\log D_\phi(Y)] - \mathbb{E}_{Y \sim G_\theta}[\log(1 - D_\phi(Y))]. \quad (5.12)$$

The structure of SeqGAN is visualized in Figure 5.20.

The generator network, G_θ , is trained to maximize the expected end reward over all possible actions y_1 given the current state s_0 ,

$$J(\theta) = \mathbb{E}[R_T | s_0, \theta] = \sum_{y_1 \in \mathcal{Y}} G_\theta(y_1 | s_0) \cdot Q_{D_\phi}^{G_\theta}(s_0, y_1). \quad (5.13)$$

Since the discriminator network D_ϕ only gives an output when the whole sequence is finished, SeqGAN applies Monte Carlo search that samples future possible actions from the vocabulary. A rollout policy is used for generating the remaining time periods. In this example, the rollout policy is the current best generator.

Network Structure

The generator network is a recurrent neural network (RNN) that uses Long-Short Memory Units (LSTMs) as building blocks. In addition to all the possible combinations of features, three special tokens are added to the dictionary as possible actions to take by the generator: <GO>, <PAD> and <END>. The input goes through several hidden layers before being fed into a softmax layer to output the probabilities of taking each possible action. A maximum sequence length of 50 is allowed before an <END> is appended to the end of each sequence.

The discriminator network is a convolutional neural network (CNN). For each sequence, we apply a series of convolutional kernels with different sizes to it, and then take the max-pooling over all outputs as the feature. The final layer is a fully connected layer with sigmoid as the activation function.

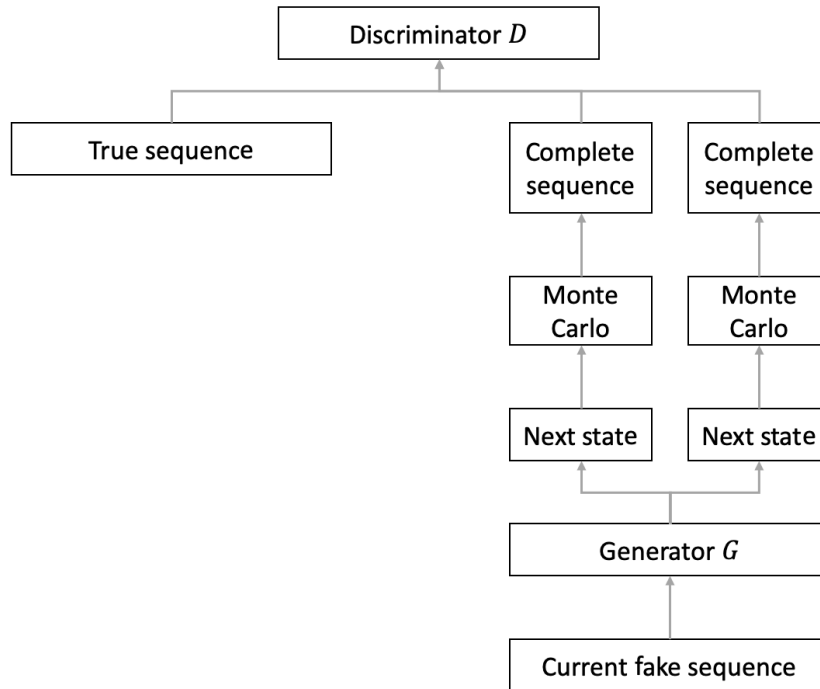


Figure 5.20: Structure of SeqGAN

Hyperparameter Settings. For this example, we use mostly the default settings in the original SeqGAN paper. On the generator side, the embedding dimension is 32, the hidden dimension is 100 and the maximum sequence length is 50. The batchsize is 64. On the discriminator side, the embedding dimension is 32, the kernels' sizes are [1,2,3,4,5,6,7,8,9,10,15], the number of kernels are [100, 200, 200, 200, 200, 100, 100, 100, 100, 100, 160], the dropout rate is 0.25, the L^2 regularization parameter is 0.2, and the batchsize is 64.

Evaluation of results

The similarity between generated records and the historical true records is evaluated using the following criteria:

1. Marginal distribution of each individual features;
2. Dependency of each pair of features, measured by the correlation coefficients;
3. Stability of accuracy of machine learning models on the generated and the true datasets.

Marginal distributions. We check the marginal distributions of each variable. Since they are all categorical, we plot out the pie charts that show the percentage of every possible category. Below are some samples:

Figure 5.21 shows the marginal distribution of the `action` variable;

Figure 5.22 shows the marginal distribution of the `indication_of_collateralization` variable;

Figure 5.23 shows the marginal distribution of the `price_forming` variable;

Figure 5.24 compares the marginal distribution of the `rounded_notional_amount` variable.

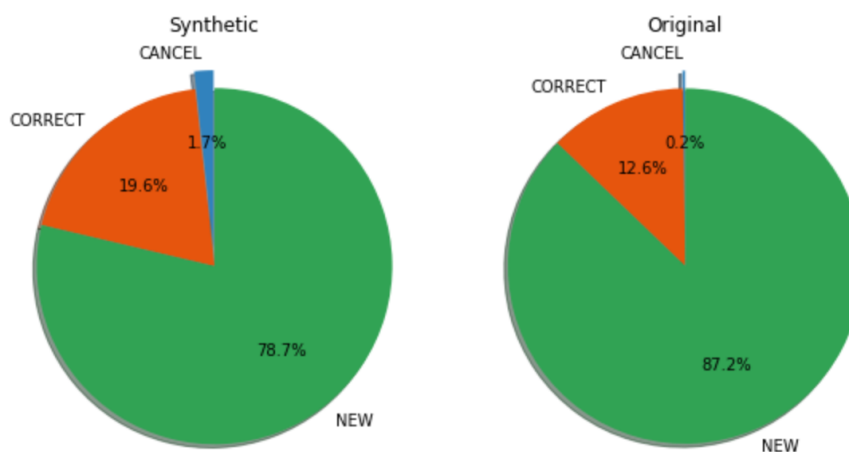


Figure 5.21: Pie chart of the action variable on the generated and the original datasets

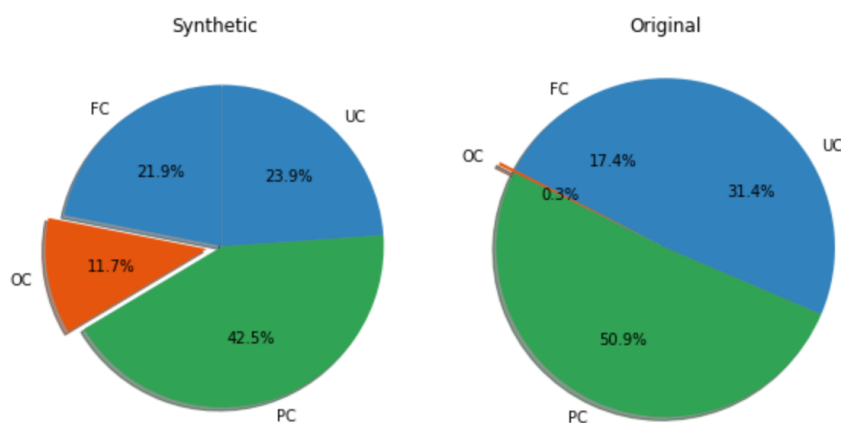


Figure 5.22: Pie chart of the collateral variable on the generated and the original datasets

Pairwise correlations. In addition to the marginal distributions, we also check the pairwise relationship between variables; that is, if the simulated dataset could reconstruct the correlations between variables. In Figure 5.25, red cells denote positive correlations between variables, while blue cells denote negative correlations. We can see that SeqGAN is able to reconstruct the directions of 15 out of 21 pairwise correlations.

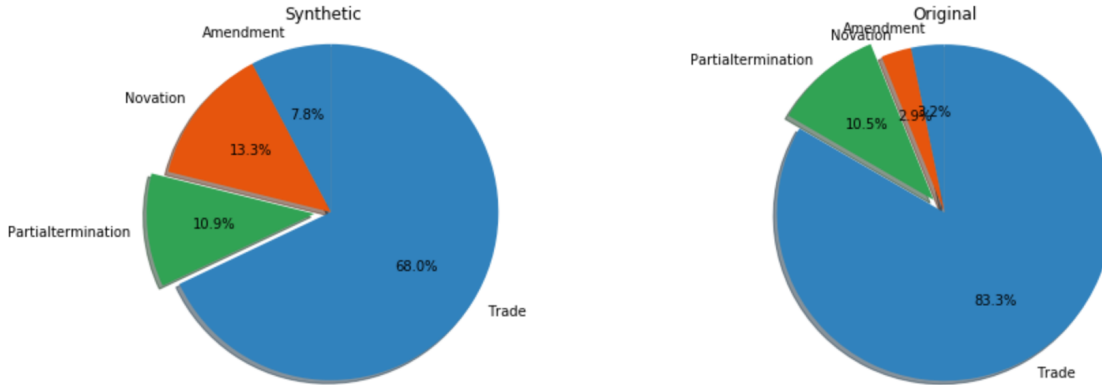


Figure 5.23: Pie chart of the price forming variable on the generated and the original datasets

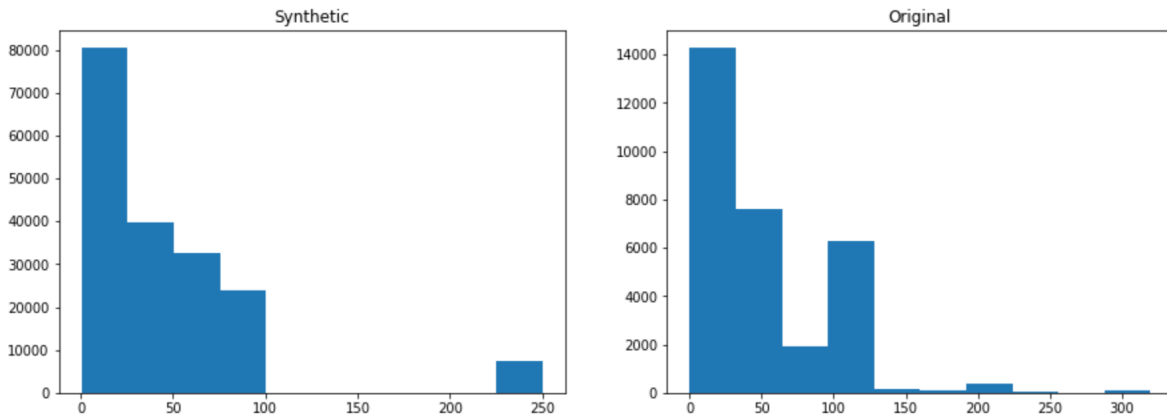


Figure 5.24: Histograms of the rounded notional amount variable on the generated and the original datasets

Stability of machine learning algorithms. In practice, after the new dataset is generated, we train various machine learning models. The generated new dataset is the training set and the real dataset becomes the test set. Ideally, we would observe that the accuracies of the models trained on the synthetic data remain roughly the same on the real dataset. In the model selection setting, however, a weaker sense of stability is required; that is, we would expect that the better performing model on the generated dataset should still perform better in the original dataset. To measure this kind of stability and to check whether the simulated dataset recovers the joint distribution of all variables, we introduce the idea of Synthetic Ranking Agreement (SRA) from Jordon, Yoon, and Schaar (2018).

Formally, let $\mathcal{A}_1, \dots, \mathcal{A}_k$ be the k models considered for selection. For each dataset \mathcal{D} , we split it into training set \mathcal{D}_1 and testing set \mathcal{D}_2 . Let m be a performance metric, e.g., accuracy, and let \mathcal{D}^G be the synthetic dataset generated by the generator G . Then we would expect the following relationship: if for two models \mathcal{A}_i and \mathcal{A}_j , $1 \leq i, j \leq k$, \mathcal{A}_i performs better than

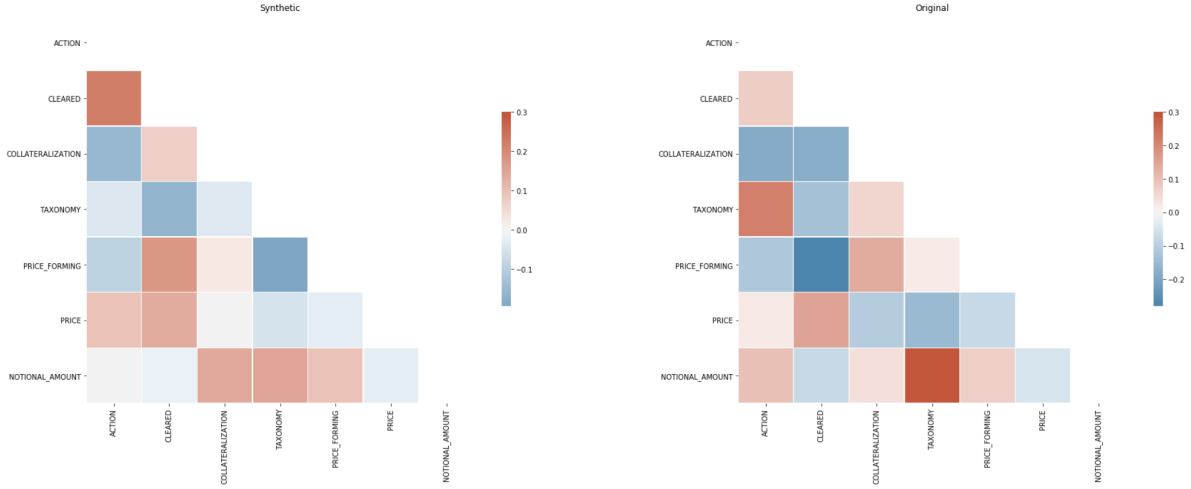


Figure 5.25: Heatmaps of the pairwise correlation coefficients on the generated and the original datasets

\mathcal{A}_j on the synthetic dataset, that is,

$$m(\mathcal{A}_i(\mathcal{D}_1^G), \mathcal{D}_2^G) < m(\mathcal{A}_j(\mathcal{D}_1^G), \mathcal{D}_2^G), \quad (5.14)$$

then it also performs better on the original dataset, that is

$$m(\mathcal{A}_i(\mathcal{D}_1), \mathcal{D}_2) < m(\mathcal{A}_j(\mathcal{D}_1), \mathcal{D}_2). \quad (5.15)$$

To come up with a metric for a generator G , the synthetic ranking agreement score is defined as

$$\text{SRA}(G) = \frac{1}{k(k-1)} \sum_{i=1}^k \sum_{j \neq i} 1_{((R_i - R_j)(S_i - S_j) > 0)}. \quad (5.16)$$

where $R_i = m(\mathcal{A}_i(\mathcal{D}_1), \mathcal{D}_2)$, $S_i = m(\mathcal{A}_i(\mathcal{D}_1^G), \mathcal{D}_2^G)$.

In this example, we consider the following 5 machine learning models, which are all very popular in the data science community:

1. Random Forest,
2. Logistic Regression,
3. Linear Support Vector Machine (SVM),
4. Multinomial Naive Bayes,
5. Gaussian Naive Bayes.

The models are all trained directly out-of-the-box from the Scikit-learn package with default hyperparameters and configurations. For each variable in the dataset, we use it as the label and the rest of the variables as features. We split the samples into 80% for training and 20% for testing. Table 5.6 shows the model ranks for the `rank` variable. The SRA score is 0.90. Random Forest is the best model. The rest are not as good, but the performances on both datasets are similar.

	Random Forest	Logistic Reg.	Linear SVM	Multinomial NB	Gaussian NB
Original	0.872	0.693	0.728	0.653	0.664
Synthetic	0.968	0.766	0.752	0.553	0.682

Table 5.6: Accuracy of models predicting the cleared variable on the original and the synthetic datasets

When predicting the `price_formulation` variable using other variables, the results are shown in Table 5.7. The SRA score is 0.90. Random forest remains the best.

	Random Forest	Logistic Reg.	Linear SVM	Multinomial NB	Gaussian NB
Original	0.901	0.857	0.823	0.784	0.653
Synthetic	0.916	0.752	0.753	0.586	0.512

Table 5.7: Accuracy of models predicting the `price_formulation` variable on the original and the synthetic datasets

When we predict the price of CDS using other variables, the models yield the following results in Table 5.8. The SRA score is 0.90.

	Random Forest	Logistic Reg.	Linear SVM	Multinomial NB	Gaussian NB
Original	0.941	0.929	0.929	0.789	0.867
Synthetic	0.958	0.697	0.623	0.545	0.591

Table 5.8: Accuracy of models predicting the price variable on the original and the synthetic datasets

From the results above, we conclude that the best performing model among all 5, the random forest, has similar performance on the original dataset and the synthetic one. The other 4 models, on the other hand, could have different predictive powers on both datasets, but their relative rankings stay the same. This shows that our synthetic dataset is capable of running model selection tasks.

Limitations and Future Works

In this example, we only consider fixed lengths of order sequences. In fact, it could be better if the length of order sequences are dynamically determined, based on factors such as densities of incoming orders or volatilities of the market.

To ensure a finite action space, we divide the range of the continuous features into bins. Although this is understandable since the variables themselves are truncated at the beginning, the number and sizes of bins are arbitrary for now.

Finally, the rollout policy for the discriminator is chosen to be the current best generator, whose value is slow to evaluate. To improve training speed, the rollout policy could be chosen as a separate model with less complexity.

Conclusion

In this section of the dissertation, we show the following three facts:

- The Sequential GAN model is capable of learning the underlying distribution of the CDX dataset and simulating new order records based on this information;
- The synthetic dataset, the marginal distributions, and pairwise correlations among features are well-kept;
- Machine learning model selections done on the new dataset are consistent with the original one.

Bibliography

- [1] Jinwon An and Sungzoon Cho. “Variational autoencoder based anomaly detection using reconstruction probability”. In: *Special Lecture on IE 2* (2015), pp. 1–18.
- [2] Grigory Antipov, Moez Baccouche, and Jean-Luc Dugelay. “Face aging with conditional generative adversarial networks”. In: *2017 IEEE International Conference on Image Processing*. IEEE. 2017, pp. 2089–2093.
- [3] Antreas Antoniou, Amos Storkey, and Harrison Edwards. “Data augmentation generative adversarial networks”. In: *arXiv preprint arXiv:1711.04340* (2017).
- [4] Martin Arjovsky. “Towards principled methods for training generative adversarial networks”. In: *ArXiv Preprint: 1701.04862* (2017).
- [5] Martin Arjovsky, Soumith Chintala, and Léon Bottou. “Wasserstein generative adversarial networks”. In: *International Conference on Machine Learning*. 2017, pp. 214–223.
- [6] Arindam Banerjee, Xin Guo, and Hui Wang. “On the optimality of conditional expectation as a Bregman predictor”. In: *IEEE Transactions on Information Theory* 51.7 (2005), pp. 2664–2669.
- [7] Arindam Banerjee, Srujana Merugu, Inderjit S Dhillon, and Joydeep Ghosh. “Clustering with Bregman divergences”. In: *Journal of Machine Learning Research* 6.Oct (2005), pp. 1705–1749.
- [8] Heinz H Bauschke and Jonathan M Borwein. “On projection algorithms for solving convex feasibility problems”. In: *SIAM review* 38.3 (1996), pp. 367–426.
- [9] Güzin Bayraksan and David K Love. “Data-driven stochastic programming using phi-divergences”. In: *The Operations Research Revolution*. INFORMS, 2015, pp. 1–19.
- [10] Aharon Ben-Tal, Dick Den Hertog, Anja De Waegenaere, Bertrand Melenberg, and Gijs Rennen. “Robust solutions of optimization problems affected by uncertain probabilities”. In: *Management Science* 59.2 (2013), pp. 341–357.
- [11] Aharon Ben-Tal, Laurent El Ghaoui, and Arkadi Nemirovski. *Robust optimization*. Vol. 28. Princeton University Press, 2009.
- [12] Aharon Ben-Tal and Arkadi Nemirovski. “Robust convex optimization”. In: *Mathematics of operations research* 23.4 (1998), pp. 769–805.

- [13] David Berthelot, Thomas Schumm, and Luke Metz. “Began: Boundary equilibrium generative adversarial networks”. In: *arXiv preprint arXiv:1703.10717* (2017).
- [14] Daniele Bianchi, Matthias Büchner, and Andrea Tamoni. “Bond risk premia with machine learning”. In: *USC-INET Research Paper* 19-11 (2019).
- [15] Jose Blanchet, Lin Chen, and Xun Yu Zhou. “Distributionally robust mean-variance portfolio selection with Wasserstein distances”. In: *ArXiv Preprint: 1802.04885* (2018).
- [16] Jose Blanchet, Yang Kang, and Karthyek Murthy. “Robust Wasserstein profile inference and applications to machine learning”. In: *ArXiv Preprint: 1610.05627* (2016).
- [17] Jose Blanchet, Henry Lam, Qihe Tang, and Zhongyi Yuan. “Applied robust performance analysis for actuarial applications”. In: *Technical Report*. the Society of Actuaries (SOA), 2017.
- [18] Konstantinos Bousmalis, Nathan Silberman, David Dohan, Dumitru Erhan, and Dilip Krishnan. “Unsupervised pixel-level domain adaptation with generative adversarial networks”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 3722–3731.
- [19] Christopher Bowles, Liang Chen, Ricardo Guerrero, Paul Bentley, Roger Gunn, Alexander Hammers, David Alexander Dickie, Maria Valdés Hernández, Joanna Wardlaw, and Daniel Rueckert. “GAN augmentation: augmenting training data using generative adversarial networks”. In: *arXiv preprint arXiv:1810.10863* (2018).
- [20] Lev M Bregman. “The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming”. In: *USSR Computational Mathematics and Mathematical Physics* 7.3 (1967), pp. 200–217.
- [21] Yann Brenier. “Polar factorization and monotone rearrangement of vector-valued functions”. In: *Communications on Pure and Applied Mathematics* 44.4 (1991), pp. 375–417.
- [22] Andreas Buja, Werner Stuetzle, and Yi Shen. “Loss functions for binary class probability estimation and classification: Structure and applications”. In: *Working draft, November 3* (2005).
- [23] Luis A Caffarelli. “Some regularity properties of solutions of Monge Ampere equation”. In: *Communications on Pure and Applied Mathematics* 44.8-9 (1991), pp. 965–969.
- [24] Luis A Caffarelli. “The regularity of mappings with a convex potential”. In: *Journal of the American Mathematical Society* 5.1 (1992), pp. 99–104.
- [25] Yair Censor and Arnold Lent. “An iterative row-action method for interval convex programming”. In: *Journal of Optimization theory and Applications* 34.3 (1981), pp. 321–353.

- [26] Sourav Chatterji, Ichitaro Yamazaki, Zhaojun Bai, and Jonathan A Eisen. “Compost-Bin: A DNA composition-based algorithm for binning environmental shotgun reads”. In: *Annual International Conference on Research in Computational Molecular Biology*. Springer. 2008, pp. 17–28.
- [27] Luyang Chen, Markus Pelger, and Jason Zhu. “Deep learning in asset pricing”. In: *Available at SSRN 3350138* (2019).
- [28] Shibing Chen and Alessio Figalli. “Partial W_2 , p regularity for optimal transport maps”. In: *Journal of Functional Analysis* 272.11 (2017), pp. 4588–4605.
- [29] Xi Chen, Yan Duan, Rein Houthoofd, John Schulman, Ilya Sutskever, and Pieter Abbeel. “Infogan: Interpretable representation learning by information maximizing generative adversarial nets”. In: *Advances in Neural Information Processing Systems*. 2016, pp. 2172–2180.
- [30] Michael Collins, Robert E Schapire, and Yoram Singer. “Logistic regression, AdaBoost and Bregman distances”. In: *Machine Learning* 48.1-3 (2002), pp. 253–285.
- [31] Briec Conan-Guez and Fabrice Rossi. “Multi-layer perceptrons for functional data analysis: a projection based approach”. In: *International Conference on Artificial Neural Networks*. Springer. 2002, pp. 667–672.
- [32] AR De Pierro and AN Iusem. “A relaxed version of Bregman’s method for convex programming”. In: *Journal of Optimization Theory and Applications* 51.3 (1986), pp. 421–440.
- [33] T De Wet. “Goodness-of-fit tests for location and scale families based on a weighted L_2 -Wasserstein distance measure”. In: *Test* 11.1 (2002), pp. 89–107.
- [34] Erick Delage and Yinyu Ye. “Distributionally robust optimization under moment uncertainty with application to data-driven problems”. In: *Operations research* 58.3 (2010), pp. 595–612.
- [35] Emily L Denton, Soumith Chintala, and Rob Fergus. “Deep generative image models using a laplacian pyramid of adversarial networks”. In: *Advances in Neural Information Processing Systems*. 2015, pp. 1486–1494.
- [36] V Dobrić and Joseph E Yukich. “Asymptotics for transportation cost in high dimensions”. In: *Journal of Theoretical Probability* 8.1 (1995), pp. 97–118.
- [37] Roland L Dobrushin. “Perturbation methods of the theory of Gibbsian fields”. In: *Lectures on Probability Theory and Statistics*. Springer, 1996, pp. 1–66.
- [38] Roland L Dobrushin. “Prescribing a system of random variables by conditional distributions”. In: *Theory of Probability & Its Applications* 15.3 (1970), pp. 458–486.
- [39] Peyman Mohajerin Esfahani and Daniel Kuhn. “Data-driven distributionally robust optimization using the Wasserstein metric: Performance guarantees and tractable reformulations”. In: *Mathematical Programming* 171.1-2 (2018), pp. 115–166.

- [40] Lawrence Craig Evans and Ronald F Garipey. *Measure Theory and Fine Properties of Functions*. Chapman and Hall/CRC, 2015.
- [41] Guanhao Feng, Jingyu He, and Nicholas G Polson. “Deep learning for predicting asset returns”. In: *arXiv preprint arXiv:1804.09314* (2018).
- [42] 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.
- [43] Maayan Frid-Adar, Eyal Klang, Michal Amitai, Jacob Goldberger, and Hayit Greenspan. “Synthetic data augmentation using GAN for improved liver lesion classification”. In: *2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018)*. IEEE, 2018, pp. 289–293.
- [44] Béla A Frigyik, Santosh Srivastava, and Maya R Gupta. “Functional Bregman divergence and Bayesian estimation of distributions”. In: *IEEE Transactions on Information Theory* 54.11 (2008), pp. 5130–5139.
- [45] Rui Gao and Anton J Kleywegt. “Distributionally robust stochastic optimization with Wasserstein distance”. In: *arXiv preprint arXiv:1604.02199* (2016).
- [46] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. “Learning to forget: Continual prediction with LSTM”. In: (1999).
- [47] Laurent El Ghaoui, Maksim Oks, and Francois Oustry. “Worst-case value-at-risk and robust portfolio optimization: A conic programming approach”. In: *Operations research* 51.4 (2003), pp. 543–556.
- [48] Corrado Gini. *Sulla misura della concentrazione e della variabilità dei caratteri*. Pre-miate officine grafiche C. Ferrari, 1914.
- [49] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. “Generative adversarial nets”. In: *Advances in Neural Information Processing Systems*. 2014, pp. 2672–2680.
- [50] Shihao Gu, Bryan T Kelly, and Dacheng Xiu. “Autoencoder Asset Pricing Models”. In: *Available at SSRN* (2019).
- [51] Shihao Gu, Bryan Kelly, and Dacheng Xiu. *Empirical asset pricing via machine learning*. Tech. rep. National Bureau of Economic Research, 2018.
- [52] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. “Improved training of Wasserstein GANs”. In: *Advances in Neural Information Processing Systems*. 2017, pp. 5767–5777.
- [53] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. “beta-VAE: Learning basic visual concepts with a constrained variational framework”. In: *International Conference on Learning Representations*. Vol. 3. 2017.

- [54] Sepp Hochreiter and Jürgen Schmidhuber. “Long short-term memory”. In: *Neural computation* 9.8 (1997), pp. 1735–1780.
- [55] Blanka Horvath, Aitor Muguruza, and Mehdi Tomas. “Deep Learning Volatility”. In: *Available at SSRN 3322085* (2019).
- [56] Zhaolin Hu and L Jeff Hong. “Kullback-Leibler divergence constrained distributionally robust optimization”. In: *Available at Optimization Online* (2013).
- [57] John C Hull and Alan D White. “Valuation of a CDO and an n-th to default CDS without Monte Carlo simulation”. In: *The Journal of Derivatives* 12.2 (2004), pp. 8–23.
- [58] John C Hull and Alan D White. “Valuing credit default swaps I: No counterparty default risk”. In: *The Journal of Derivatives* 8.1 (2000), pp. 29–40.
- [59] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. “Image-to-image translation with conditional adversarial networks”. In: *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. 2017, pp. 1125–1134.
- [60] Ruiwei Jiang and Yongpei Guan. “Data-driven chance constrained stochastic program”. In: *Mathematical Programming* 158.1-2 (2016), pp. 291–327.
- [61] Lee K Jones and Charles L Byrne. “General entropy criteria for inverse problems, with applications to data compression, pattern classification, and cluster analysis”. In: *IEEE Transactions on Information Theory* 36.1 (1990), pp. 23–30.
- [62] James Jordon, Jinsung Yoon, and Mihaela van der Schaar. “Measuring the quality of Synthetic data for use in competitions”. In: *arXiv preprint arXiv:1806.11345* (2018).
- [63] Tarja Joro, Anne R Niu, and Paul Na. “A simulation-based First-to-Default (FTD) Credit Default Swap (CDS) pricing approach under jump-diffusion”. In: *Proceedings of the 36th Conference on Winter Simulation*. Winter Simulation Conference. 2004, pp. 1632–1636.
- [64] Anatoli Juditsky, Arkadi Nemirovski, and Claire Tauvel. “Solving variational inequalities with stochastic mirror-prox algorithm”. In: *Stochastic Systems* 1.1 (2011), pp. 17–58.
- [65] Dongwan D Kang, Jeff Froula, Rob Egan, and Zhong Wang. “MetaBAT, an efficient tool for accurately reconstructing single genomes from complex microbial communities”. In: *PeerJ* 3 (2015), e1165.
- [66] Dongwan Kang, Feng Li, Edward S Kirton, Ashleigh Thomas, Rob S Egan, Hong An, and Zhong Wang. “MetaBAT 2: an adaptive binning algorithm for robust and efficient genome reconstruction from metagenome assemblies”. In: *PeerJ Preprints* 7 (2019), e27522v1.
- [67] Leonid Vasilevich Kantorovich and Gennady S Rubinstein. “On a space of completely additive functions”. In: *Vestnik Leningrad. Univ* 13.7 (1958), pp. 52–59.
- [68] Leonid Kantorovitch. “On the translocation of masses”. In: *Management Science* 5.1 (1958), pp. 1–4.

- [69] Diederik P Kingma and Jimmy Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014).
- [70] Diederik P Kingma and Max Welling. “Auto-encoding variational Bayes”. In: *arXiv preprint arXiv:1312.6114* (2013).
- [71] Jyrki Kivinen and Manfred K Warmuth. “Relative loss bounds for multidimensional regression problems”. In: *Advances in Neural Information Processing Systems*. 1998, pp. 287–293.
- [72] Naveen Kodali, Jacob Abernethy, James Hays, and Zsolt Kira. “How to train your DRAGAN”. In: *arXiv preprint arXiv:1705.07215* 2.4 (2017).
- [73] Lutz Krause, Naryttza N Diaz, Alexander Goesmann, Scott Kelley, Tim W Nattkemper, Forest Rohwer, Robert A Edwards, and Jens Stoye. “Phylogenetic classification of short environmental DNA fragments”. In: *Nucleic Acids Research* 36.7 (2008), pp. 2230–2239.
- [74] Viveka Kulharia, Arnab Ghosh, Amitabha Mukerjee, Vinay Namboodiri, and Mohit Bansal. “Contextual RNN-GANs for abstract reasoning diagram generation”. In: *Thirty-First AAAI Conference on Artificial Intelligence*. 2017.
- [75] Matt J Kusner, Brooks Paige, and José Miguel Hernández-Lobato. “Grammar variational autoencoder”. In: *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org. 2017, pp. 1945–1954.
- [76] John Lafferty. “Additive models, boosting, and inference for generalized divergences”. In: *In Proceedings of 12th Annual Conference on Computational Learning Theory*. Citeseer. 1999.
- [77] Henry Lam and Enlu Zhou. “Quantifying uncertainty in sample average approximation”. In: *Proceedings of the 2015 Winter Simulation Conference*. IEEE Press. 2015, pp. 3846–3857.
- [78] Jean-Michel Lasry and Pierre-Louis Lions. “Mean field games”. In: *Japanese Journal of Mathematics* 2.1 (2007), pp. 229–260.
- [79] Guy Le Besnerais, J-F Bercher, and Guy Demoment. “A new look at entropy for solving linear inverse problems”. In: *IEEE Transactions on Information Theory* 45.5 (1999), pp. 1565–1578.
- [80] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. “Photo-realistic single image super-resolution using a generative adversarial network”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 4681–4690.
- [81] Chuan Li and Michael Wand. “Precomputed real-time texture synthesis with markovian generative adversarial networks”. In: *European Conference on Computer Vision*. Springer. 2016, pp. 702–716.

- [82] Jerry Li, Aleksander Madry, John Peebles, and Ludwig Schmidt. “Towards understanding the dynamics of generative adversarial networks”. In: *arXiv preprint arXiv:1706.09884* (2017).
- [83] Xiaopeng Li and James She. “Collaborative variational autoencoder for recommender systems”. In: *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM. 2017, pp. 305–314.
- [84] Mario Lucic, Olivier Bachem, and Andreas Krause. “Strong coresets for hard and soft Bregman clustering with applications to exponential family mixtures”. In: *arXiv preprint arXiv:1508.05243* (2015).
- [85] CL Mallows et al. “A note on asymptotic joint normality”. In: *The Annals of Mathematical Statistics* 43.2 (1972), pp. 508–515.
- [86] Xudong Mao, Qing Li, Haoran Xie, Raymond YK Lau, Zhen Wang, and Stephen Paul Smolley. “Least squares generative adversarial networks”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 2794–2802.
- [87] Giovanni Mariani, Florian Scheidegger, Roxana Istrate, Costas Bekas, and Cristiano Malossi. “Bagan: Data augmentation with balancing gan”. In: *arXiv preprint arXiv:1803.09655* (2018).
- [88] Lars Mescheder, Sebastian Nowozin, and Andreas Geiger. “Adversarial variational bayes: Unifying variational autoencoders and generative adversarial networks”. In: *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org. 2017, pp. 2391–2400.
- [89] Marcial Messmer. “Deep learning and the cross-section of expected returns”. In: *Available at SSRN 3081555* (2017).
- [90] Paul Milgrom and Ilya Segal. “Envelope theorems for arbitrary choice sets”. In: *Econometrica* 70.2 (2002), pp. 583–601.
- [91] Mehdi Mirza and Simon Osindero. “Conditional generative adversarial nets”. In: *arXiv preprint arXiv:1411.1784* (2014).
- [92] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. “Asynchronous methods for deep reinforcement learning”. In: *International Conference on Machine Learning*. 2016, pp. 1928–1937.
- [93] Monzoorul Haque Mohammed, Tarini Shankar Ghosh, Nitin Kumar Singh, and Sharmila S Mande. “SPHINX - an algorithm for taxonomic binning of metagenomic sequences”. In: *Bioinformatics* 27.1 (2010), pp. 22–30.
- [94] Gaspard Monge. *Mémoire sur la théorie des déblais et des remblais*. De l’Imprimerie Royale, 1781.

- [95] Axel Munk and Claudia Czado. “Nonparametric validation of similar distributions and assessment of goodness of fit”. In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 60.1 (1998), pp. 223–241.
- [96] Noboru Murata, Takashi Takenouchi, Takafumi Kanamori, and Shinto Eguchi. “Information geometry of U-Boost and Bregman divergence”. In: *Neural Computation* 16.7 (2004), pp. 1437–1481.
- [97] Hongseok Namkoong and John C Duchi. “Stochastic gradient methods for distributionally robust optimization with f-divergences”. In: *Advances in Neural Information Processing Systems*. 2016, pp. 2208–2216.
- [98] Arkadi Nemirovski. “Prox-method with rate of convergence $O(1/t)$ for variational inequalities with Lipschitz continuous monotone operators and smooth convex-concave saddle point problems”. In: *SIAM Journal on Optimization* 15.1 (2004), pp. 229–251.
- [99] Arkadii Semenovich Nemirovski. *Problem Complexity and Method Efficiency in Optimization*. Wiley, 1983.
- [100] Dong Nie, Roger Trullo, Jun Lian, Caroline Petitjean, Su Ruan, Qian Wang, and Dinggang Shen. “Medical image synthesis with context-aware generative adversarial networks”. In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer. 2017, pp. 417–425.
- [101] Jakob Nybo Nissen, Casper Kaae Sonderby, Jose Juan Almagro Armenteros, Christopher Heje Groenbech, Henrik Bjorn Nielsen, Thomas Nordahl Petersen, Ole Winther, and Simon Rasmussen. “Binning microbial genomes using deep learning”. In: *BioRxiv* (2018), p. 490078.
- [102] Augustus Odena, Christopher Olah, and Jonathon Shlens. “Conditional image synthesis with auxiliary classifier gans”. In: *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org. 2017, pp. 2642–2651.
- [103] MC Pardo and Igor Vajda. “About distances of discrete distributions satisfying the data processing theorem of information theory”. In: *IEEE transactions on Information Theory* 43.4 (1997), pp. 1288–1293.
- [104] MC Pardo and Igor Vajda. “On asymptotic properties of information-theoretic divergences”. In: *IEEE Transactions on Information Theory* 49.7 (2003), pp. 1860–1867.
- [105] Yuval Peres. “Mixing for Markov chains and spin systems”. In: *Unpublished notes* (2005). URL: www.stat.berkeley.edu/~peres/ubc.pdf.
- [106] Thomas Nordahl Petersen, Oksana Lukjancenko, Martin Christen Frølund Thomsen, Maria Maddalena Sperotto, Ole Lund, Frank Møller Aarestrup, and Thomas Sicheritz-Pontén. “MGmapper: reference based mapping and taxonomy annotation of metagenomics sequence reads”. In: *PLoS One* 12.5 (2017), e0176469.

- [107] Warren B Powell. “A unified framework for optimization under uncertainty”. In: *Optimization Challenges in Complex, Networked and Risky Systems*. INFORMS, 2016, pp. 45–83.
- [108] Yunchen Pu, Zhe Gan, Ricardo Henao, Xin Yuan, Chunyuan Li, Andrew Stevens, and Lawrence Carin. “Variational autoencoder for deep learning of images, labels and captions”. In: *Advances in Neural Information Processing Systems*. 2016, pp. 2352–2360.
- [109] Svetlozar T Rachev. *Probability Metrics and the Stability of Stochastic Models*. John Wiley & Sons Ltd., Chichester, 1991.
- [110] Svetlozar T Rachev and Ludger Ruschendorf. *Mass Transportation Problems: Volume I: Theory*. Vol. 1. Springer Science & Business Media, 1998.
- [111] Alec Radford, Luke Metz, and Soumith Chintala. “Unsupervised representation learning with deep convolutional generative adversarial networks”. In: *arXiv preprint arXiv:1511.06434* (2015).
- [112] Scott Reed, Zeynep Akata, Xinchun Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. “Generative adversarial text to image synthesis”. In: *arXiv preprint arXiv:1605.05396* (2016).
- [113] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. “Stochastic back-propagation and approximate inference in deep generative models”. In: *arXiv preprint arXiv:1401.4082* (2014).
- [114] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. “Improved techniques for training gans”. In: *Advances in Neural Information Processing Systems*. 2016, pp. 2234–2242.
- [115] Herbert E Scarf. *A min-max solution of an inventory problem*. Tech. rep. RAND CORP SANTA MONICA CALIF, 1957.
- [116] Stanislau Semeniuta, Aliaksei Severyn, and Erhardt Barth. “A hybrid convolutional variational autoencoder for text generation”. In: *arXiv preprint arXiv:1702.02390* (2017).
- [117] Madhvi Sethi, Parthiv Thakkar, and Zahid M Jamal. “A Simulation Model for Pricing the Spread in a Credit Default Swap: Application and Analysis”. In: *SDMIMD Journal of Management* 9.2 (2018), pp. 9–18.
- [118] Soroosh Shafieezadeh-Abadeh, Peyman Mohajerin Esfahani, and Daniel Kuhn. “Distributionally robust logistic regression”. In: *Advances in Neural Information Processing Systems*. 2015, pp. 1576–1584.

- [119] Hoo-Chang Shin, Neil A Tenenholtz, Jameson K Rogers, Christopher G Schwarz, Matthew L Senjem, Jeffrey L Gunter, Katherine P Andriole, and Mark Michalski. “Medical image synthesis for data augmentation and anonymization using generative adversarial networks”. In: *International Workshop on Simulation and Synthesis in Medical Imaging*. Springer. 2018, pp. 1–11.
- [120] Herbert Spohn. *Large Scale Dynamics of Interacting Particles*. Springer Science & Business Media, 2012.
- [121] Santosh Srivastava, Maya R Gupta, and Béla A Frigyik. “Bayesian quadratic discriminant analysis”. In: *Journal of Machine Learning Research* 8.Jun (2007), pp. 1277–1305.
- [122] Michel Talagrand. “Concentration of measure and isoperimetric inequalities in product spaces”. In: *Publications Mathématiques de l’Institut des Hautes Etudes Scientifiques* 81.1 (1995), pp. 73–205.
- [123] Hiroshi Tanaka. “An inequality for a functional of probability distributions and its application to Kac’s one-dimensional model of a Maxwellian gas”. In: *Probability Theory and Related Fields* 27.1 (1973), pp. 47–52.
- [124] Ben Taskar, Simon Lacoste-Julien, and Michael I Jordan. “Structured prediction, dual extragradient and Bregman projections”. In: *Journal of Machine Learning Research* 7.Jul (2006), pp. 1627–1653.
- [125] T. Tieleman and G. Hinton. “Lecture 6.5-RMSProp: divide the gradient by a running average of its recent magnitude”. In: *COURSERA: Neural Networks for Machine Learning* 4.2 (2012).
- [126] Duy Tin Truong, Eric A Franzosa, Timothy L Tickle, Matthias Scholz, George Weingart, Edoardo Pasolli, Adrian Tett, Curtis Huttenhower, and Nicola Segata. “MetaPhlAn2 for enhanced metagenomic taxonomic profiling”. In: *Nature Methods* 12.10 (2015), p. 902.
- [127] Bart PG Van Parys, Peyman Mohajerin Esfahani, and Daniel Kuhn. “From Data to Decisions: Distributionally Robust Optimization is Optimal”. In: *arXiv preprint arXiv:1704.04118* (2017).
- [128] Leonid Nisonovich Vaserstein. “Markov processes over denumerable products of spaces, describing large systems of automata”. In: *Problemy Peredachi Informatsii* 5.3 (1969), pp. 64–72.
- [129] Cédric Villani. *Optimal Transport: Old and New*. Vol. 338. Springer Science & Business Media, 2008.
- [130] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. “Generating videos with scene dynamics”. In: *Advances in Neural Information Processing Systems*. 2016, pp. 613–621.

- [131] A. Wibisono and A. C. Wilson. “A variational perspective on accelerated methods in optimization”. In: *Proceedings of the National Academy of Sciences of the United States of America*. 2016.
- [132] Jelmer M Wolterink, Tim Leiner, Max A Viergever, and Ivana Išgum. “Generative adversarial networks for noise reduction in low-dose CT”. In: *IEEE Transactions on Medical Imaging* 36.12 (2017), pp. 2536–2545.
- [133] Raymond A Yeh, Chen Chen, Teck Yian Lim, Alexander G Schwing, Mark Hasegawa-Johnson, and Minh N Do. “Semantic image inpainting with deep generative models”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 5485–5493.
- [134] Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. “Seqgan: Sequence generative adversarial nets with policy gradient”. In: *Thirty-First AAAI Conference on Artificial Intelligence*. 2017.
- [135] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. “Self-attention generative adversarial networks”. In: *arXiv preprint arXiv:1805.08318* (2018).
- [136] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, and Dimitris N Metaxas. “Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 5907–5915.
- [137] Junbo Zhao, Michael Mathieu, and Yann LeCun. “Energy-based generative adversarial network”. In: *arXiv preprint arXiv:1609.03126* (2016).
- [138] Ding Zhou, Jia Li, and Hongyuan Zha. “A new mallows distance based metric for comparing clusterings”. In: *Proceedings of the 22nd International Conference on Machine learning*. ACM. 2005, pp. 1028–1035.
- [139] Jun-Yan Zhu, Philipp Krähenbühl, Eli Shechtman, and Alexei A Efros. “Generative visual manipulation on the natural image manifold”. In: *European Conference on Computer Vision*. Springer. 2016, pp. 597–613.
- [140] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. “Unpaired image-to-image translation using cycle-consistent adversarial networks”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 2223–2232.
- [141] Xinyue Zhu, Yifan Liu, Zengchang Qin, and Jiahong Li. “Data augmentation in emotion classification using generative adversarial networks”. In: *arXiv preprint arXiv:1711.00648* (2017).