Game-Theoretic Approaches for Generative Modeling

DISSERTATION

Submitted in Partial Fulfillment of
the Requirements for
the Degree of

DOCTOR OF PHILOSOPHY (Computer Science)

at the

NEW YORK UNIVERSITY
TANDON SCHOOL OF ENGINEERING

by

Jian Gao

January 2020

Approved by the Guidance Committee:

Major: Computer Science

Tembine Hamidou
Assistant Professor of
Electrical and Computer Engineering
12 - 06 - 2019
Date

Edward Wong
Associate Professor of
Computer Science and Engineering
12/6/2019
Date

Julian Togelius
Associate Professor of
Computer Science and Engineering
12/6/2019
Date

Gus Xia
Assistant Research Professor of
Computer Science and Engineering
12/6/2019
Date

Vita

Jian Gao comes from the beautiful city of Shanghai, China. He received the bachelor degree in Automation and the master degree in Pattern Recognition & Intelligent Systems from Tongji University in 2012 and 2015.

In September 2015, Jian Gao was enrolled in the PhD program in Computer Science at Tandon School of Engineering, New York University. Since then, he joined the Learning & Game Theory Laboratory and began working with Professor Hamidou Tembine on game theory and deep learning projects. He worked as a research intern in Philips Lighting North America Corporation in Summer 2018, and in Hitachi America Ltd in Summer 2019.

His research interest involves machine learning, computer vision, speech processing and game theory. During the PhD study, Jian Gao has published 14 papers and presented at 8 international conferences. In 2017, he won the Best Paper Award at the International Conference on Wireless Networks and Mobile Communications.

Jian Gao served as peer reviewer of several IEEE Transactions and international conferences, including IEEE Conference on Decision and Control, IEEE PES General Meeting, IEEE Transactions on Vehicular Technology, and IEEE Transactions on Intelligent Transportation Systems.

Acknowledgements

I would like to thank my advisor Prof. Hamidou Tembine for his long-timed support. I enjoyed every discussion with him, whether it was about technical programming issues or mathematical problems in game theory and machine learning. His insightful guidance gave me a chance to work on many wonderful projects during my PhD study. His optimistic enthusiasm and the spirit of exploration helped to guide and push me completing this thesis and many other works. During my whole PhD, I am very grateful to have the freedom to explore various research areas. The time and impact from him will last for long on shaping my flavor of scientific research as well as the way I live.

I would also like to thank all of my collaborators and coauthors, and the many students and post-docs in the lab, in particular Yida Xu, Julian Barreiro-Gomez, Massa Ndong, Michalis Smyrnakis, Meng Wang, Guoxian Dai, Lin Xu, Jin Xie, Jin Zhu, Fan Zhu and Jin Xie. We learned much from each other's work and experienced numerous inspiring conversations about life and study. I am grateful for having worked with great people at NYU and NYUAD. Thanks also to Erin Anderson as our director of graduate and postdoctoral affairs.

I want to thank my class teachers, in particular Prof. Edward Wong and Prof. Davi Geiger, the most talented and enthusiastic teachers I had. Finally and most essentially, I am grateful to my parents, who made me pursue a research path and support me all the way.

Jian Gao

January 2020

To Zeng Zuoxiang, Xue Weilan, my parents

To all the Ph.D. pursuing brave souls

ABSTRACT

Game-Theoretic Approaches for Generative Modeling

by

Jian Gao

Advisor: Prof. Tembine Hamidou, Ph.D.

Submitted in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy (Computer Science)

January 2020

Artificial intelligence has achieved great success in the past decade, but there still remains a big gap between machine and human intelligence. Most supervised approaches rely on large-scale manually labeled data, and are vulnerable to domain shifts and adversarial attacks. To achieve collective intelligence, an alternative is to build a generic model of the world. It enables an agent to reason its environment before making decisions. Discriminative models learn the conditional distribution of class label from data, while generative models include the full distribution of data themselves. It requires deeper understanding of the nature. When the environment

is completely known, a rational agent acts toward the most desirable outcome; when there is uncertainty, it acts to optimize the expected payoff.

This dissertation outlines the problem of generative modeling and proposes several solutions from a game theory perspective. We begin by reviewing the concepts of strategic and cooperative games, including Nash games, Bayesian games, and robust games.

In the first part, we introduce three commonly used metrics to measure the discrepancy between probability distributions: Bregman divergence, f-divergence, and Wasserstein distance. We compare their properties for generative modeling and show some approximation algorithms for computation. This part is the basis of the dissertation as many algorithms use these metrics to model the uncertainty.

In the second part, we study generative models for data synthesis and propose the concept of distributionally robust games (DRG). These are multi-player games in which each player has to make decision in an uncertain environment. It leads to a distributionally robust optimization problem, and the solution is reached by dynamically optimizing the worst-case payoff. We formulate the problem by the statistical notions of f-divergence and Wasserstein distance, and develop learning algorithms to solve the Nash equilibria. Then, we use the DRG framework to train deep generative models and propose DRGAN. Motivated by the recent progress in computer vision, we apply it to the challenging task of unsupervised image generation.

In the third part, we focus on generative models for data manipulation. We develop a conditional generative model for emotional speech conversion: EmoGAN. It enables the transfer of emotion-related characteristics of a speech signal while preserving the speaker's identity and linguistic content. We assume that the

speech signal can be decomposed into an emotion-invariant content code and an emotion-related style code in latent space. Emotion conversion is performed by extracting and recombining the content code of the source speech and the style code of the target emotion. An autoencoder model is designed to learn the disentangled representations, in which the encoders, decoders and emotion classifiers form a DRG with competitive and collaborative agent coalitions. We test this approach on a nonparallel corpora with four emotions and evaluate the generated speech.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In the last decade, artificial intelligence has achieved remarkable success in a wide range of domains. Recent algorithms beat humans on tasks such as image classification [1, 2], object detection [3, 4], natural language understanding [5, 6], playing games [7, 8], and approaching the human ability in many other fields [9, 10]. However, there is still a big gap between machine and human intelligence.

Most successful approaches are based on supervised learning. It relies on a set of labeled data provided by a knowledgeable external supervisor, e.g., human annotators. The system learns to extrapolate from training data to testing data by assuming they are drawn from the same distribution. Annotating millions of samples is expensive. What's worse is, the models developed for specific tasks and trained on specific datasets may perform poorly in other similar situations.

Instead of huge training sets, humans can easily learn new concepts from very few examples and have much stronger generalization ability. One guess is that we may have some pre-trained model encoded in our genes, which has been trained for millions of years. Another more reasonable explanation is that we learn knowledge

by interacting with the environment, relying more on our own experience than on external supervisors. Not only humans but also animals follow this strategy.

To pursue real intelligence, an alternative is to build a generic model of the world. An intelligent agent should explore the environment to learn this model as well as exploit this model to make decisions. In reinforcement learning, agents sense the environment, take actions, and get feedback to refine their strategies. This approach alleviates the addiction to labeled data. But there may exist significant uncertainty, either from corrupted observations or from the nonstationary environment. Supervised learning handles the uncertainty based on manually labeled data, which often has a unimodal probability distribution. Therefore it can be approximated by a Gaussian whose mean value is used for deterministic predictions. However, unsupervised data often represents a more general multimodal distribution. In this case, the agent needs to learn how to model the uncertainty and make stochastic predictions.

Learning a general model of the environment is more desirable than focusing on an isolated task. Unsupervised learning aims to find the underlying structure in a collection of unlabeled samples. Density estimation explicitly models the data distribution, and generative models mimic the data generating process such that new synthesized samples follow the same distribution as the original ones. It first learns a generically effective representation of the input, and then use it for data synthesis and other applications. Before the deep learning revolution, traditional machine learning systems involve a procedure called "feature engineering" to extract useful features from data, and then perform classification on those low-dimensional feature descriptors. It is an expensive manual process which requires expert knowledge to create those hand-crafted features [11, 12]. Nowadays, many neural network

models [2, 13, 14, 15] can automatically learn such feature representations. Though these models are not adequate to recover the original data distribution, they can be useful parts in training intelligent agents.

This dissertation investigates the problem of training intelligent agents in dynamic environments with uncertainty and learning generative models from a game-theory perspective. Generally speaking, the standard decision theory (Single-Agent Reinforcement Learning) focuses on single-agent systems, where there is only one agent in a defined environment. The system state is solely subject to the actions of one agent. Most of machine learning research is concerned with a single agent tackling a task, by optimizing the reward of a goal-directed agent in a stationary environment, using supervised learning. However, many real world problems involve a large group of agents interacting with an uncertain environment.

Game theory (Multi Agent Reinforcement Learning, or Distributed strategic learning) focuses on multi-agent systems where the environment changes as the result of the set of joined actions taken by all agents in the system. The uncertainty is not only inherent in the domain but also comes from the unpredictable actions of other agents. For example in a traffic network, the decision of any vehicle is the result of the road condition as well as the behavior of many other vehicles on the road. Typically the complexity of multi-agent systems increases with the number of agents involved.

Another issue is how the agents act with each other. Game-theoretic models may exhibit competition, cooperation, or mixed behaviors among agents. In filtering problems, agents act as a group (particle swarm) to reason about the state space. In adversarial machine learning, a pair of agents compete against each other to improve the overall system performance. In distributed learning, agents make

individual decisions based on their local observations and neighborhood messages, with no need to infer the policies of others [16].

Compared with single-agent systems, game-theoretic models offer better representations of many cognitive activities in the real world. Since these models are dealing with more complex scenarios, a number of issues need to be addressed. There are three major challenges when implementing multi-agent systems:

- **Curse of dimensionality**: As pointed out by Richard E. Bellman [17] in dynamic optimization, when the dimension of the state space increases, the volume of space increases so fast that available data getting sparse, and the complexity of dynamic programming equations grows exponentially. For instance, the distributionally robust optimization problem [18] involves naturally the space of probability distributions which is of infinite dimensions for continuous action spaces. Many models designed for certain game environments fail as the number of agents increases. Mean-field learning [19] could be useful for analyzing large interacting systems.

- **Model the uncertainty**: In multi-agent systems, the transition of an individual depends on the state of the environment as well as other agents. Due to the dynamic environment conditions, unpredictable agent behavior and corrupted observation, modeling the uncertainty is a major issue for learning multi-agent systems. Many problems do not have an optimal solution because of the uncertainty involved in the environment's dynamics. Robust optimization [20] is a common model for games with incomplete information.

- **Stabilize the training process**: Training across multiple agents with different objectives is another nightmare. Some problems offer convergence

toward global optima, while others may be trapped in local optima or never converge. Take for example the two-player zero-sum games, some objective functions will not converge using gradient descent. The model parameters oscillate when implementing the minimax algorithm. Alternative objective functions and learning algorithms are required to stabilize the training process.

We will explore those problems and present several game-theoretic models for multi-agent learning systems. In the first part, we review several commonly used metrics to measure the discrepancy between probability distributions. Three distance metrics will be studied: Bregman divergence, $f-$divergence, and Wasserstein distance. We compare their properties for learning generative models, and introduce some methods to approximate the Wasserstein distance. This part is the basis of other sections as many algorithms are designed using them as the objective functions.

In the second part, we investigate generative models for data synthesis. The objective is to learn an unknown data distribution from a set of samples drawn from it. We formulate the problem as a distributionally robust game (DRG) in which the uncertain environment acts as an adversarial player against the agent by always providing the worst-case scenario. This leads to a distributionally robust optimization (DRO) problem, in which the unknown data distribution is learned by dynamically optimizing the worst-case payoff. We develop Bregman learning algorithms to solve the robust Nash equilibria. Theoretical findings are illustrated in a convex setting and its limitations are tested with a non-convex non-concave objective. We also design practical implementations to train deep generative models, and test it on data clustering and image synthesis.

In the third part, we focus on generative models for data manipulation (style

transfer). We introduce a conditional generative model in speech processing. It enables the transfer of emotion-related characteristics of a speech signal while preserving the speaker's identity and linguistic content. We assume that the speech signal can be decomposed into an emotion-invariant content code and an emotion-related style code in latent space. Emotion conversion is performed by extracting and recombining the content code of the source speech and the style code of the target emotion. The disentangled representations of domain-specific style information and domain-invariant content information are modeled by autoencoders and domain classifiers. The encoders, decoders and emotion classifiers form a (semi-cooperative game) with competitive and collaborative agent coalitions. We test this approach on a nonparallel corpora with four emotions and evaluate the generated speech. Experiment results show that our approach can effectively change the emotions and keep high voice quality.

Thesis Outline The rest of this thesis is organized as follows. Chapter 2 reviews the previous work on game theory and generative modeling. Chapter 3 investigates several widely used divergence concepts and objective functions for learning generative models. Chapter 4 introduces the distributionally robust game, in which the agents' strategies are learnt by optimizing their worst-case performance. Chapter 5 develops learning algorithms to solve the distributionally robust Nash equilibrium under Wasserstein metric. The proposed approach is applied to both shallow and deep generative models. Chapter 6 proposes a nonparallel voice conversion model to learn the disentangled representations for human speech. It can automatically change the emotion conveyed in speech signals and does not rely on paired data. Chapter 7 concludes the thesis and discusses future research directions.

Chapter 2

Background

This chapter reviews some concepts related to the thesis. In section 2.1, we introduce the background of game theory. It involves the definition of Nash equilibrium and the concepts of many different types of games, including Nash game, Bayesian game, robust game, and cooperative game. In section 2.2, we give a brief summary of the recent progress in generative modeling and figure out the pros and cons for each kind of model.

2.1 Background of Game Theory

Game theory (launched by Von Neumann [21], followed by John Nash [22]) is a theoretical framework that uses mathematical tools to model and analysis strategic interaction between rational agents. Standard machine learning (and decision theory) study scenarios with a single decision maker, while game theoretic models involve multiple agents with different goals, in which the action of each agent will affect the outcome of others. The foundations of game theory were launched by John von Neumann and Oskar Morgenstern, in their famous book *The*

Theory of Games and Economic Behavior [21] published in 1944. The theory was then developed by John Nash in his PhD thesis on non-cooperative games [23], containing the definition and properties of Nash Equilibrium. Since then, game theory has been widely developed and applied to a broad range of fields, such as wireless communication [19, 24], traffic network [25], filtering [26, 27] and data assimilation [28]. In artificial intelligence, it offers a way for goal-directed learning from interaction, which is different from other machine learning approaches that focus on learning from external supervisors.

Game theory can be divided into two subfields [29]: strategic games in which agents act independently or competitively, with each player tries to maximize its own profit, and cooperative games where agents form coalitions and enforce coordinated actions to optimize the overall behavior of a team. We begin by introducing some important concepts in game theory.

2.1.1 Representation of Games

In game theory, a well-defined game has four essential elements [30]: the players of the game, the information available to each player, the action profile at each decision point, and the payoff for each outcome. Most noncooperative games are defined in strategic and extensive forms, while cooperative games are mainly presented in characteristic function form.

Strategic form This is the standard representation of a game, which is also known as normal form. A game in strategic form is defined by the set of players, the strategy spaces and the payoff functions of each player.

Definition 2.1. *(Strategic Form Representation)*
A game in strategic form is defined as an ordered triple $\mathcal{G} = (\mathcal{N}, (\mathcal{A}_j)_{j\in\mathcal{N}}, (u_j)_{j\in\mathcal{N}})$, where:

- $\mathcal{N} = \{1, 2, ..., n\}$ is a finite set of players.
- $\mathcal{A}_j$ is the set of actions for player j. Let $\mathcal{A} = \mathcal{A}_1 \times \mathcal{A}_2 \times \cdots \times \mathcal{A}_n$
- $u_j : \mathcal{A} \to \mathbb{R}$ is the payoff (utility) function for player j.

In this definition, the set of actions for players can be infinite, and we will discuss the infinite dimension problem in games with continuous action spaces. The payoff function for player j depends not only on its own action a_j but also on other players' actions $a_{-j} := (a_1, \cdots, a_{j-1}, a_{j+1}, \cdots, a_n)$, that is, the outcome of interactive decisions of all players.

When $n = 2$, the strategic form is usually represented by a payoff matrix. One player chooses the row and the other chooses the column. The intersection of each row and column shows the payoff of their corresponding strategies. People can easily find whether the strategies are stable by looking at the matrix entries. Some classic strategic two-player games are listed below. For example in the Prisoner's Dilemma game, when Player I chooses to *Deny* and Player II chooses to *Confess*, the payoff for Player I is -10 (10 years in prison), and the payoff for Player II is 0.

Example 2.2 (*Strategic Games*)**.**

Prisoner's Dilemma

		Player II	
		Confess	*Deny*
Player I	*Confess*	$-8, -8$	$0, -10$
	Deny	$-10, 0$	$-1, -1$

Battle of the Sexes

		Woman	
		Football	*Concert*
Man	*Football*	$2, 1$	$0, 0$
	Concert	$0, 0$	$1, 2$

Inspection Game

		Employee	
		Work	*Shirk*
Boss	*Inspect*	$30, 20$	$10, 15$
	Not Inspect	$45, 20$	$5, 35$

Extensive form Extensive form is often used to describe sequential games, where the actions of players are implemented sequentially rather than simultaneously, and the latter player has some information about earlier actions. A game in extensive form is represented by a multi-player decision tree (see Figure 2.1), in which each vertex is a decision point and each edge indicates a possible move for a player. The payoffs are given by the leaf nodes, which are terminal points of the game. Backward induction is used to determine the optimal sequential moves. The extensive form can also describe games with imperfect information by introducing the notion of information set. It is a set of decision points that the players cannot distinguish where they are based on their limited information, for example, in poker and bridge.

Characteristic function form The characteristic function is generally used to describe cooperative games, by listing a value for each coalition. This form was originally proposed by John von Neumann [21]. In cooperative games, when a coalition $\mathcal{C}$ is formed, it works against the remaining part $\mathcal{N} \mid \mathcal{C}$ as two individuals

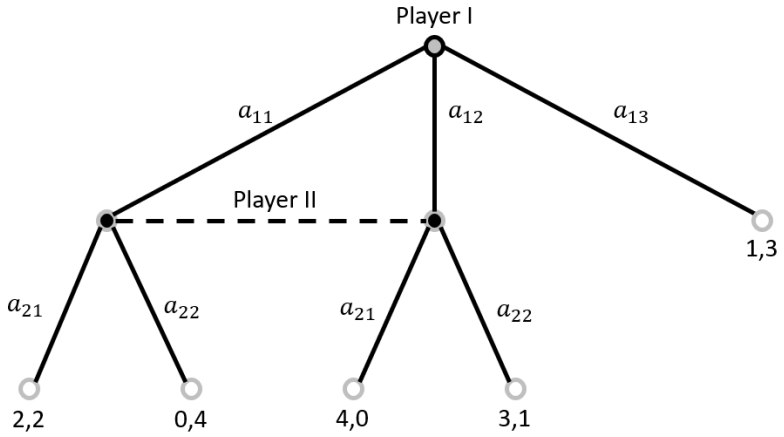

Figure 2.1: An example of game in extensive form. Player I moves first by choosing from actions $\{a_{11}, a_{12}, a_{13}\}$, and then followed by Player II's choice from $\{a_{21}, a_{22}\}$. The dotted line represents the information set of player 2, indicating a game of imperfect information. The sequential equilibrium is (3,1).

were playing a strategic game. The characteristic function specifies the payoff of each coalition, while separate rewards are not given.

Definition 2.3. *(Cooperative Games)*

A cooperative game in characteristic function form is given by specifying a value for every coalition. Formally, the game is defined by a pair $(\mathcal{N}; v)$, where:

- $\mathcal{N} = \{1, 2, ..., n\}$ is a finite set of players. A subset of $\mathcal{N}$ is called a coalition. The set of all possible coalitions is denoted by $2^{\mathcal{N}}$.
- $v : 2^{\mathcal{N}} \to \mathbb{R}$ is the characteristic function that specifies the minimum value a set of players can gain by forming a coalition when playing against all other players. By convention the value of null set is zero $v(\varnothing) = 0$.

Take Prisoner's Dilemma for instance, if we denote the two suspects as A, B, and let the policeman be the third player C, then the characteristic function lists the value for each coalition (see Table 2.1).

	$v(\varnothing) = 0$	
$v(A) = -8$	$v(B) = -8$	$v(C) = 2$
$v(AB) = -2$	$v(AC) = 8$	$v(BC) = 8$
	$v(ABC) = 0$	

Table 2.1: Prisoner's Dilemma as a coalition game.

2.1.2 Solution Concepts

Once a game is described, people may expect to know what outcomes will ensue for rational players. We will introduce two solution concepts *Nash equilibrium* and *Maxmin Value*, capturing different behavioral aspects in a game. The first is an expression of stability, and the second reflects the notion of security.

Nash Equilibrium Stability is an important principle to predict the behavior of a rational agent. For instance in the Prisoner's Dilemma (Example 2.2), *Confess* is always a better strategy than *Deny*. (*Confess*, *Deny*) is not stable because for player II, *Confess* will bring a better payoff (-1) than *Deny* (-10). (*Deny*, *Deny*) is also not stable because each player has a motivation to choose *Confess* for a payoff increase from -1 to 0.

The concept of *Nash Equilibrium* was proposed by John Nash [22] in 1950s. In game theory, each player acts to its best possible outcome regarding the actions of other players. At equilibrium, no player can increase the payoff by changing only its own strategy.

Definition 2.4. *(Nash Equilibrium)*
Let $\mathcal{G} = (\mathcal{N}, (\mathcal{S}_j)_{j\in\mathcal{N}}, (u_j)_{j\in\mathcal{N}})$ be a game with n players, where S_j is the strategy profile for player j, $S = S_1 \times S_2 \times \cdots \times S_n$ is the set of strategy profiles for all players, and $u(s) = (u_1(s), \cdots, u_n(s))$ is the payoff function evaluated at $s \in S$. Let

s_j be a strategy for player j and s_{-j} be a vector of strategies for all other players. Note that the payoff $u_j(s_j, s_{-j})$ depends on the strategies chosen by player j as well as the strategies chosen by all other players. A strategy vector $s^* = (s_1^*, \cdots, s_n^*)$ is a *Nash Equilibrium (NE)* if for each player $j \in \mathcal{N}$ and each strategy $s_j \in S_j$, the following condition is satisfied:

$$u_j(s^*) \geq u_j(s_j, s_{-j}^*) \tag{2.1}$$

The payoff vector $u(s^*)$ is the equilibrium payoff corresponding to the Nash equilibrium s^*.

That is to say, no single player has a profitable deviation at the Nash equilibrium. This seems to be a general requirement for any solution concept, otherwise the expected result will be changed by at least one player for better reward. Much research of game theory has been devoted to studying equilibria and their properties. However, Nash Equilibrium is not the final solution. Some games have no NE, and others may have infinite number of equilibria. Even the unique equilibrium may not constitute a good recommendation. For example in Prisoner's Dilemma (Example 2.2), the NE payoff (-8,-8) is a bad outcome for both players.

Maxmin Value Sometimes, Nash equilibrium does not show the expected behavior for rational agents. Consider the game in Example 2.5: (W, D) is the unique NE with payoff $(5, 5)$, but it's dangerous for Bob if Sam chooses C by mistake. Thus Bob may prefer a safer strategy U since it has a guaranteed payoff 3, though less than the NE payoff 5. On the other side, if Sam realizes this, he will also flee to a safer strategy C, which in turn increases Bob's motivation to avoid the dangerous outcome -99.

Example 2.5 (*Risky Nash Equilibrium*)**.**

		Sam		$\min u_{Bob}$
		C	D	
Bob	U	$3, 1$	$3, -15$	3
	V	$4, 0$	$-9, 1$	-9
	W	$-99, 2$	$5, 5$	-99
$\min u_{Sam}$		0	-15	$(3, 0)$

This example shows another aspect of rational behavior: intelligent agent prefers some alternative solution concept than NE to avoid risky outcomes, where a good payoff is guaranteed based only on her own efforts, without relying on the behavior of other players. The last column shows the worst possible outcomes for Bob when he chooses a specific strategy. So does the bottom line for Sam.

Definition 2.6. *(Maxmin Strategy and Maxmin Value)*
In a game $\mathcal{G} = (\mathcal{N}, (\mathcal{S}_j)_{j\in\mathcal{N}}, (u_j)_{j\in\mathcal{N}})$, let $s_j \in \mathcal{S}_j$ be a strategy for player j, and $s_{-j} \in \mathcal{S}_{-j}$ be the strategies for other players. The *maxmin strategy* for player j is

$$\arg\max_{s_j} \min_{s_{-j}} u_j(s_j, s_{-j}), \tag{2.2}$$

which maximizes her worst-case payoff in case all other players happen to choose strategies that give her the biggest loss. The *maxmin value* (or security level) for player j is

$$\underline{u}_j := \max_{s_j} \min_{s_{-j}} u_j(s_j, s_{-j}), \tag{2.3}$$

which is the minimum amount of payoff guaranteed by a *maxmin strategy* disregarding the rational behavior of other players.

In Example 2.5, the maxmin value of Bob is 3 with maxmin strategy U, and

the maxmin value of Sam is 0 with maxmin strategy C. When both of them choose maxmin strategies, the outcome $(3, 1)$ is greater than Sam's maxmin value.

Moreover, when the strategy set is infinite, we replace the maximum and minimum in Definition 2.6 with the supremum and infimum:

$$\underline{u}_j := \sup_{s_j \in \mathcal{S}_j} \inf_{s_{-j} \in \mathcal{S}_{-j}} u_j(s_j, s_{-j}) \tag{2.4}$$

The maxmin strategy is well defined when the payoff function is continuous and the strategy domain is compact.

2.1.3 Strategic Games

Traditional game theory studies the behavior of individual players and predicts their actions and payoffs by analyzing Nash equilibrium. In strategic games, players cannot form coalitions and must compete independently. Cooperative games can be expressed in strategic game frameworks by assuming that coalitions have the ability to enforce coordinated behavior, given all possible strategies available to players due to the external enforcement of cooperation.

Information is a crucial part in decision making. A game is one of complete information if each player knows the strategies and payoffs available to the others. Most game theoretic models study games with imperfect or incomplete information.

In this section, we will review the previous work in games with complete information (Nash Games), games with incomplete information (Bayesian Games), and games with payoff uncertainty (Robust Games). Before that, we start with an important class: the two-player zero-sum games, where the solution concepts of *Nash equilibrium* and *Maxmin value* coincide. This solution concept fulfills both

the goals of stability and security.

2.1.3.1 Two-player zero-sum games

Two-player games are defined by the set of players $\mathcal{N} = \{I, II\}$, strategies $\mathcal{S} = \{\mathcal{S}_I, \mathcal{S}_{II}\}$ and utilities $u = \{u_I(s_I, s_{II}), u_{II}(s_I, s_{II})\}$. A two-player game is a zero-sum game if for each pair of strategies, the total sum of the two players' payoff is zero.

$$u_I(s_I, s_{II}) + u_{II}(s_I, s_{II}) = 0 \tag{2.5}$$

An example is given below, it is a strict competitive game where one wins exactly the amount the other's lose.

Example 2.7 (*Two-player zero-sum game*)**.**

		Player II	
		a_{21}	a_{22}
Player I	a_{11}	$A, -A$	$B, -B$
	a_{12}	$C, -C$	$D, -D$

Table 2.2: A general form of two-player zero-sum games

Two-player zero-sum games were the first class of games studied mathematically and yield formal results in the early stage of game theory. Most classical games fall into this category, such as chess, Go and tennis. Poker and gambling are popular examples in its extended class of multi-player constant-sum games. Decision problems can also be modeled as two-player zero-sum games, in which the decision maker is one player and the environment is the other, who controls the uncertain parameters to minimize the decision maker's payoff. Since the players in zero-sum games act diametrically opposed against each other, there is no room for

cooperation, which focuses the study on isolated parts of the games such as equilibrium, information flows, and modeling the uncertainty.

Let u be the payoff function of the game, which is Player I's gain and Player II's loss: $u_I = u, u_{II} = -u$. Player I tries to maximize the gain, and its maxmin value is

$$\underline{u}_I = \max_{s_I \in \mathcal{S}_I} \min_{s_{II} \in \mathcal{S}_{II}} u(s_I, s_{II})$$

Player II tries to minimize the loss, and its maxmin value is

$$\underline{u}_{II} = \max_{s_{II} \in \mathcal{S}_{II}} \min_{s_I \in \mathcal{S}_I} (-u(s_I, s_{II})) = - \min_{s_{II} \in \mathcal{S}_{II}} \max_{s_I \in \mathcal{S}_I} u(s_I, s_{II})$$

Denote

$$\begin{aligned} \underline{v} &:= \max_{s_I \in \mathcal{S}_I} \min_{s_{II} \in \mathcal{S}_{II}} u(s_I, s_{II}) \\ \bar{v} &:= \min_{s_{II} \in \mathcal{S}_{II}} \max_{s_I \in \mathcal{S}_I} u(s_I, s_{II}), \end{aligned} \tag{2.6}$$

where $\underline{v}$ is the maxmin value and $\bar{v}$ is the minmax value of the game. It is guaranteed that Player I can receive payment at least $\underline{v}$, and Player II will pay at most $\bar{v}$. We can prove that the inequality $\underline{v} \leq \bar{v}$ always holds. In addition, the *minimax theorem* [31] provides conditions that the inequality is also an equality.

Theorem 2.1 (*von Neumann's minimax theorem*)**.**

Let $\mathcal{S}_I \subset \mathbb{R}^m$ and $\mathcal{S}_{II} \subset \mathbb{R}^n$ be compact convex sets. If $u : \mathcal{S}_I \times \mathcal{S}_{II} \to \mathbb{R}$ is a continuous function that convex-concave, i.e. $u(s_I, \cdot) : \mathcal{S}_{II} \to \mathbb{R}$ is concave for fixed s_I, and $u(\cdot, s_{II}) : \mathcal{S}_I \to \mathbb{R}$ is convex for fixed s_{II}, then:

$$\max_{s_I \in \mathcal{S}_I} \min_{s_{II} \in \mathcal{S}_{II}} u(s_I, s_{II}) = \min_{s_{II} \in \mathcal{S}_{II}} \max_{s_I \in \mathcal{S}_I} u(s_I, s_{II}) \tag{2.7}$$

Definition 2.8. For a two-player game, if the equality holds, the quantity $v :=$

$\underline{v} = \bar{v}$ is called the *value* of the game, and the maxmin and minmax strategies are called *optimal strategies*.

Every zero-sum games with an equilibrium has a value. The two solution concepts Nash equilibrium and minmax value coincide to the same result, making the goals of stability and security unified.

Theorem 2.2. *In a two-player zero-sum game, if $s^* = (s_I^*, s_{II}^*)$ is a Nash equilibrium, then it has a value $v = u(s_I^*, s_{II}^*)$, and the strategies s_I^*, s_{II}^* are optimal strategies. On the other hand, if the game has a value v and optimal strategies s_I^*, s_{II}^*, then $s^* = (s_I^*, s_{II}^*)$ is a Nash equilibrium with payoff $(v, -v)$.*

It means, Nash equilibrium is equivalent to the value of the game, and equilibrium strategies are optimal strategies. Moreover, there is a geometric interpretation for the value of a two-player zero-sum game.

Definition 2.9 (*Saddle Point*)**.**

The strategy pair (s_I^*, s_{II}^*) is a saddle point of the function $u : \mathcal{S}_I \times \mathcal{S}_{II} \to \mathbb{R}$ if

$$u(s_I^*, s_{II}^*) \geq u(s_I, s_{II}^*), \forall s_I \in \mathcal{S}_I$$
$$u(s_I^*, s_{II}^*) \leq u(s_I^*, s_{II}), \forall s_{II} \in \mathcal{S}_{II}$$

In a two-player zero-sum game, (s_I^*, s_{II}^*) is a saddle point of the function u if and only if s_I^*, s_{II}^* are optimal strategies for Player I and Player II, and $u(s_I^*, s_{II}^*)$ is the value of the game.

2.1.3.2 Games with Complete Information

In games with complete information, each player knows the strategy profile and outcome of all players. Complete information means all parameters of the

game, including every player's type, strategies, and payoff functions are common knowledge. Under the assumptions of rational players and complete information, Nash [22, 23] proved the existence of Nash equilibrium in mixed strategies.

Theorem 2.3 (*Nash's Existence Theorem*)**.**

If mixed strategies are allowed, then every game with a finite number of players in which each player has a finite number of pure strategies, has at least one Nash equilibrium.

The concept of Nash equilibrium and the existence theorem offers a methodology to predict the outcome of a game. In classical game theory, any rational player who knows the other players' strategy profiles will choose his best response to those strategies, and this is a common knowledge in the game. Therefore, all players can reach consistent and deterministic predictions of other players' behavior. The predicted overcomes fall into the set of Nash equilibria. However, this kind of prediction is not always reliable in practice. As discussed in Example 2.5, players may take actions other than the Nash equilibrium to avoid risky outcomes. In many real-world problems, players may not have complete knowledge of the game's structure, and sometimes they are uncertain about their own payoffs.

2.1.3.3 Games with Incomplete Information

A more realistic setting is that players do not have complete information of a game. The payoff functions are not available because of the unknown game structure or the private information of other agents. Harsanyi [32] introduced a concept of "type" and modeled such games with incomplete information as Bayesian Games. In Bayesian Games, each player j has a finite number of "type" $t_j \in T_j$, which corresponds to a specific payoff $u_j(s_j, s_{-j}, t_j)$. The type t_j is private information

only available to player j, and the set of types T_j is common knowledge among all players. A Bayesian Game is defined as following:

Definition 2.10. *(Bayesian Games)*

A Bayesian Game of incomplete information is defined as an ordered triple $\mathcal{G} = (\mathcal{N}, p, (\mathcal{A}_j, u_j, T_j)_{j \in \mathcal{N}})$, where:

- $\mathcal{N} = \{1, 2, ..., n\}$ is a finite set of players.
- $\mathcal{A}_j$ is the set of actions for player j. Let $\mathcal{A} = \mathcal{A}_1 \times \mathcal{A}_2 \times \cdots \times \mathcal{A}_n$
- T_j is the set of types for player j. The set of type vectors is denoted by $T = T_1 \times \ldots \times T_n$.
- $u_j : T_j \times \mathcal{A} \to \mathbb{R}$ is the payoff (utility) function for player j.
- p is the prior probability distribution over T.

It is assumed that player j with type t_j, does not know the type of other players, but has a "belief" represented by the conditional probability $p(t_{-j}|t_j)$. The players in Bayesian Games have a common knowledge of the prior probability distribution over types. With the prior distribution and his own type, each player can estimate the type of other players using the Bayes rule:

$$p(t_{-j}|t_j) = \frac{p(t_{-j}, t_j)}{p(t_j)} = \frac{p(t_{-j}, t_j)}{\sum_{t_{-j} \in T_{-j}} p(t_{-j}, t_j)} \tag{2.8}$$

2.1.3.4 Robust Games

Aghassi and Bertsimas [20] introduced a parametric distribution-free model for games with incomplete information, i.e. the robust games, in which the assumption of Harsanyi's Bayesian game model is relaxed. In Bayesian games, the distribution

over types is common knowledge; while in robust games, the players do not know any probability distribution over their payoff functions. The only common knowledge is an uncertainty set that contains all possible values of the uncertain payoff parameters. The robust game model provides an distribution-free equilibrium concept, and it has been proved in [20] that the equilibrium is guaranteed to exist when the game is finite and has bounded payoff uncertainty set.

In robust games, the players take conservative strategies to seek for a guaranteed payoff in the worst-case scenario. The worst scenario is taken with regard to the uncertainty set of payoff parameters, and the solution is found by a robust optimization approach. A Robust Game is defined by

Definition 2.11. *(Robust Games)*

A Robust Game is denoted as the sequel $\mathcal{G} = (\mathcal{N}, (\mathcal{A}_j, u_j, U_j)_{j\in\mathcal{N}})$, where:

- $\mathcal{N} = \{1, 2, ..., n\}$ is a finite set of players.
- $\mathcal{A}_j$ is the set of actions for player j. Let $\mathcal{A} = \mathcal{A}_1 \times \mathcal{A}_2 \times \cdots \times \mathcal{A}_n$
- $u_j(\alpha_j; a_j, a_{-j})$ is the payoff (utility) function for player j, which depends on a vector of parameters $\alpha_j = (\alpha_j^1, \cdots, \alpha_j^k)$, $\alpha_j \in U_j \subset \mathbb{R}^k$.
- U_j is the set of all possible values of parameters in u_j, and $U = \prod_{j\in\mathcal{N}} U_j$ denotes the entire uncertainty set of the game.

In robust games, the optimal response of player j given other players' strategies a_{-j} is obtained by maximizing its worst-case payoff $\max_{a_j\in\mathcal{A}_j} \min_{\alpha_j\in U_j} u_j(\alpha_j; a_j, a_{-j})$. The robust equilibrium is defined as following

Definition 2.12. *(Robust Optimization Equilibrium)*

In an N-player robust game, the robust optimization equilibrium is the set of

strategies $(a_1^*, \cdots, a_n^*) \in \mathcal{A}$ that satisfy

$$a_j^* \in \arg\max_{a_j \in \mathcal{A}_j} \min_{\alpha_j \in U_j} u_j(\alpha_j; a_j, a_{-j}^*), \forall j \in \mathcal{N}$$

The existence of the equilibrium is given in Theorem 2.4.

Theorem 2.4 (*Existence of Equilibrium in Robust Games*)**.** *Any N-player, non-cooperative, simultaneous-move, one-shot robust game with finite number of players and each player has finite possible actions, if the uncertainty set U is bounded and there is no private information, then the existence of the robust optimization equilibrium is guaranteed.*

2.1.4 Cooperative Games

In strategic games the players act independently by optimizing its own payoff, while in cooperative games players can form coalitions that enforce coordinated actions. The models introduced in the previous section are easily adapted to cooperative games. For example the prisonerś dilemma can be overcome by involving cooperative behavior in an iterative version of the game. Cooperative games have a finite number of players with transferable payoff, and every coalition is associated with a value representing the worth of the coalition. In a cooperative game, players exhibit competition, cooperation, and mixed behaviors.

Definition 2.13. *(Cooperative Game)*
A cooperative game with transferable utility is a pair $(\mathcal{N}, v)$ such that

- $\mathcal{N} = \{1, 2, ..., n\}$ is a finite set of players. A subset of $\mathcal{N}$ is called a coalition. The collection of all coalitions is defined as $2^{\mathcal{N}}$.

- The coalition function $v : 2^{\mathcal{N}} \to \mathbb{R}$ associates each coalition with a real number satisfying $v(\emptyset) = 0$, which is the value of the coalition. This value is the sum of gains its members can guarantee by joining the coalition, and is independent of the actions of the players outside the coalition.

- The payoff can be transferred between the players in each coalition.

Cooperative game theory provides a high-level framework in which a player may be a person, a group of people, or a nation. It focuses on the structure, strategies and payoffs on the coalition level. When a coalition is formed, all its members must cooperate with each other and take joint actions. In this thesis, cooperative games are used to study systems in which players are more abstract objects such as particles, vehicles,and neural networks.

2.2 Background of Generative Modeling

In the last decade, artificial intelligence has achieved great success with the rapid development of deep neural networks (DNNs). Recent algorithms beat humans in image recognition and natural language understanding. For example, a stream of papers[1, 5, 6, 33, 34] claiming they have surpassed human baselines on the Large Scale Visual Recognition Challenge benchmark (ImageNet) [35] and the General Language Understanding Evaluation benchmark (GLUE) [36]. Apart from recognition and classification tasks, people wish to learn the mechanism of data generation and synthesize new samples with desired properties. Discriminative learning tries to infer knowledge from data, while generative modeling aims to capture the full distribution over the given data and generate new samples. This thesis focuses on the latter problem.

In probability theory, discriminative models learn a conditional distribution $p_{data}(Y|X)$ from a training set $\{(x_1, y_1), \cdots, (x_N, y_N)\}$ of i.i.d samples. If we parameterize the model by θ, then learning is about to find the optimal θ^* such that $p_\theta(Y|X) \approx p_{data}(Y|X)$. On the other hand, generative models learn to directly model $p_{data}(X)$ (the full distribution of data) or $p_{data}(X, Y)$ (the joint distribution of data and label). The former problem is easier because Y is usually in low dimension and $p_{data}(Y|X)$ often has a single mode. The conditional distribution is a categorical when Y is discrete, and is unimodal when Y is continuous. However, generative modeling aims at learning the hidden structure of unlabeled data, which means to represent high-dimensional and multimodal distributions. It often use a low-dimensional latent variable $z \in Z$ to describe the underlying factors of variation in the data, such as the high-level semantic content and the low-level details of a particular style. For example, the factors of variation in human speech may involve the linguistic content, the speakerś accent, and background noise.

Early generative models (e.g., Restricted Boltzmann Machines (RBMs) [37] and its extensions Deep Belief Networks (DBNs) [38], Deep Boltzmann Machines (DBMs)[39]) use graph-based representations as the foundation to encode data in high-dimensional space. They are trained by maximum likelihood estimation (MLE) or energy minimization. Many modern generative models are based on this idea to learn the description of images [40, 41], text [42], and feature representations [43, 44].

Directed graphic models (i.e. Bayesian networks) factorize a distribution into a product of conditional distributions over a set of independent random variables. In this framework, the joint distribution of data x and latent code z can be factorized

as $p(x, z) = p(x|z)p(z)$. The data distribution is expressed by the marginal

$$p(x) = \int_z p(x|z)p(z) \tag{2.9}$$

The prior $p(z)$ should be easy to sample from. It can be a Gaussian or uniform distribution, or any other distribution with sufficient diversity. The generative model $p(x|z)$ produce new data by conditional sampling on the latent code. It is parameterized by deep neural networks and learnt by MLE. In contrast, the inference model $p(z|x)$ describes the underlying latent factors of data. Exact $p(z|x)$ is intractable in most directed generative models.

Undirected graphical models (i.e. Markov random fields) use a set of random variables with Markov property to represent dependencies. Boltzmann machine is a type of Markov random field with an energy function defined on its network. The energy can be converted to a probability density by using Gibbs distribution.

$$p(x) = \frac{1}{Z}\exp(-E(x)), \tag{2.10}$$

where $E(x)$ is the energy, $Z = \sum_x \exp(-E(x))$ is the normalization factor and also called the partition function. The generative model is trained to minimize the energy at positive sample points (from the training set), and increase it elsewhere. When negative examples are not available, the model will collapse to have low energy everywhere. One solution is to restrict the low-energy region by imposing a sparsity constraint, e.g., sparse coding [45] and sparse auto-encoders [46]. Another option is to produce artificial negative examples using Markov chain Monte Carlo (MCMC) sampling [47], denoising autoencoders [48], or energy-based networks [49]. However, these approaches involve either intractable inference learning or high-cost

MCMC sampling to estimate the partition function gradient.

2.2.1 Variational Autoencoders

Autoencoders are neural network models to learn compressed representations of unsupervised data. A typical autoencoder has two parts: the encoder $f : \mathcal{X} \to \mathcal{Z}$ and the decoder $g : \mathcal{Z} \to \mathcal{X}$, where $\mathcal{X}$ is the input space and $\mathcal{Z}$ is the latent space. The model is trained by minimizing the reconstruction loss

$$f, g = \arg\min \parallel x - g(f(x)) \parallel^2 \tag{2.11}$$

Since $\mathcal{Z}$ is a low-dimensional space, the latent code $f(x)$ is a compressed representation of the input x. The most important factors are encoded while the redundant details are discarded. When the latent space has enough capacity, an autoencoder learns the identity mapping and become useless. Various models have been proposed to avoid learning identity, such as sparse autoencoders, denoising autoencoders, and contractive autoencoders.

Variational Autoencoders (VAEs) [50] are generative models. Unlike classical autoencoders, VAEs are trained by minimizing a *variational bound*. Recall in directed graphic models the integral of marginal likelihood $p(x) = \int_z p(x|z)p(z)$ is intractable and the true posterior $p(z|x) = \frac{p(x|z)p(z)}{p(x)}$ is not available, therefore it cannot be differentiated. VAEs introduce a new distribution $q(z|x)$ to approximate the inference model $p(z|x)$. The variational approximate posterior is parameterized as a multivariate Gaussian

$$q_\phi(z|x) = \mathcal{N}(z; \mu_\phi(x), \sigma^2_\phi(x)I) \tag{2.12}$$

where the mean and diagonal covariance are implemented by neural networks, and ϕ is the variational parameter. If we parameterize the generative model $p(x|z)$ by θ, then VAEs derive a lower bound on the marginal likelihood of data, and the loss function $\mathcal{L}(\theta, \phi; x)$ is known as the *variational bound*.

$$\log p_\theta(x) \geq \mathcal{L}(\theta, \phi; x) = \mathbb{E}_{q_\phi(z|x)} \log p_\theta(x|z) - D_{KL}(q_\phi(z|x) \parallel p_{\mathcal{Z}}(z)) \tag{2.13}$$

The first term indicates reconstruction, and the second term is the Kullback-Leibler (KL) divergence of the approximated posterior $p_\phi(z|x)$ and the true prior $p(z)$. Since $p_{\mathcal{Z}}(z)$ is unknown, people simply assume a Gaussian prior $\mathcal{Z} \sim \mathcal{N}(0, I)$. Then the KL divergence acts as a regularizer to push the latent code toward the Gaussian prior, that is, encourages μ_ϕ to be zeros and σ_ϕ to be ones. As a result, the decoder $p_\theta(x|z)$ and encoder $q_\phi(z|x)$ are trained in opposition directions of reconstructing the data and fitting the prior.

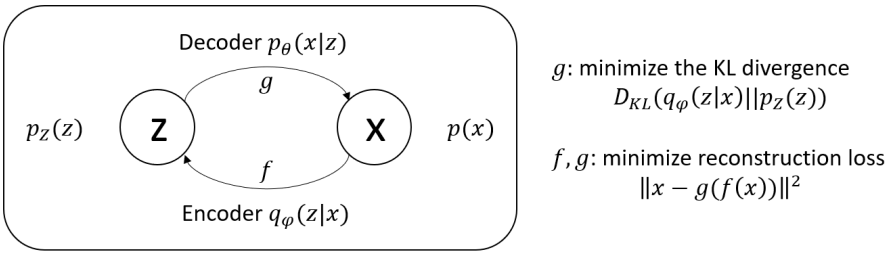

Figure 2.2: VAE as a directed graphical model. The upper line shows the generative model, and the bottom line shows variational approximation of the true inference $p_\theta(z|x)$. Parameter θ and ϕ are learnt jointly.

In probabilistic modeling, VAEs consist of a probability model of data x and latent variables z. It is a directed graphic model with joint probability $p(x, z) = p(x|z)p(z)$. New data can be generated by first sampling from the latent space $z_i \sim \mathcal{N}(0, I)$, and then convert it to a data point $x_i \sim p_\theta(x|z)$. The model is

hyper-parameter free and easy to train. However, recent researches [51, 52] find VAEs tend to generate blurry samples, which indicates unseparated modes in the reconstructed distribution.

2.2.2 Generative Adversarial Networks

Generative Adversarial Networks (GANs) [53] are implicit generative models based on game theory. Instead of explicitly model the likelihood function, GANs cast generative modeling as a game between two competing networks: a generator G and a discriminator D. The two networks play a game, in which the generator produces fake data samples that are similar to the real ones, and the discriminator's job is to distinguish them. The input of G is sampled from a trivial distribution $z \sim p_Z(z)$, and then be mapped through a differentiate network $G_{\theta_g}(z)$ to the data space. The distribution of generated samples p_g should replicate the distribution of real data p_{data}. On the other hand, D is a binary classification network $D_{\theta_d}(x)$ to separate real and fake samples. It outputs a number in $[0, 1]$, indicating the probability that the sample is real. G and D form a zero-sum game, and the objective function of GANs introduces a minimax optimization problem

$$\min_G \max_D J(D, G) = \mathbb{E}_{x \sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_Z(z)}[\log(1 - D(G(z)))] \tag{2.14}$$

Specifically, the discriminator aims to minimize the negative log-likelihood of the binary classification problem. The objective is

$$J^D(D, G) = -\mathbb{E}_{x \sim p_{data}(x)}[\log D(x)] - \mathbb{E}_{z \sim p_Z(z)}[\log(1 - D(G(z)))] \tag{2.15}$$

The generator G is trained to produce fake samples that can fool the discriminator by minimizing the *minmax objective*. In practice, Goodfellow et al. [54] suggest using an alternative loss function called the *non-saturating objective*

$$\begin{aligned} &\textbf{Minimax} && J^G(D,G) = \mathbb{E}_{z\sim p_Z(z)}[\log(1 - D(G(z)))] \\ &\textbf{Non-saturating} && J^G(D,G) = -\mathbb{E}_{z\sim p_Z(z)}[\log(D(G(z)))] \end{aligned} \tag{2.16}$$

The *minmax objective* ensures the generate samples to have low probability of being fake, while the *non-saturating objective* ensures the generate samples to have high probability of being real. Both of them have saturating problem, but in different directions (see Figure 2.3).

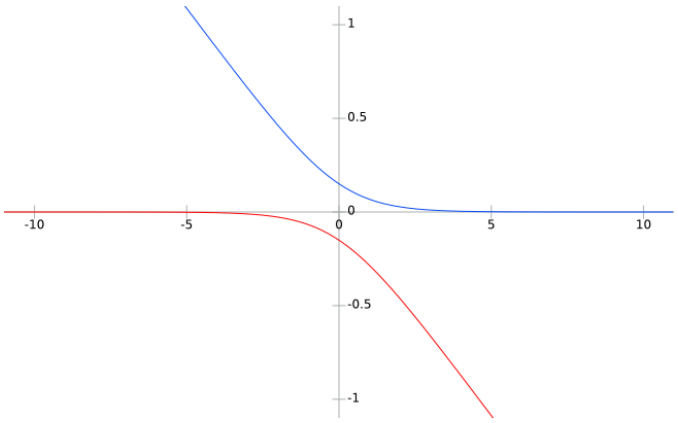

Figure 2.3: Loss function for minimax GAN (red) and non-saturating GAN (blue). $\frac{1}{2}\log(1 - D(x))$ has vanishing gradients on the left half, and $-\frac{1}{2}\log(D(x))$ has vanishing gradients on the right half, in which $D(x) = sigmoid(x)$.

Optimizing equation 2.14 means to find a *Nash equilibrium*, where G cannot be improved for fixed D, and D cannot be improved for fixed G. This is a multi-agent optimization problem. In the training process, the two networks improve each other. The model is trained when it reaches a stable point, instead of a global minimum. Theoretically, when the discriminator is completely optimized, minimizing $J(D, G)$

with respect to G is equivalent to minimizing the Jensen–Shannon (JS) divergence. Ideally, the discriminator should be completed optimized at each step before the generator updates. Goodfellow et al. [53] proved that given enough capacity, the model can reach a unique Nash equilibrium where p_g converges to p_{data}.

In practice, the model distribution p_g is limited by the generator $G_{\theta_g}(z)$, and completely optimizing D at each step is computational infeasible. Even worse, a perfect discriminator will lead to vanishing gradients (as in Figure 2.3) where the learning process get saturated. Empirically, the discriminator is updated k times for each generator's move, and k is usually less than 5.

GANs are trained by minimizing the divergence between the model and data distribution. In minimizing the JS divergence, we are trying to push the two distributions as close as possible. Thus the GAN model is encouraged to put mass only on the data manifold, at the risk of missing some patterns. In minimizing the KL divergence, we only manipulate the encoding or posterior distribution. Thus the VAEs are encouraged to put mass on the entire data manifold, at the risk of generating blurry samples.

The GAN loss is not directly calculated from data samples, but defined by a discriminative task. The discriminator can incorporate a wide variety of models and take the advantage of powerful modern neural networks. No approximate inference or MCMC sampling is needed during training. The generator $G_{\theta_g}(z)$ can be viewed as the decoder in VAE, but is deterministic with respect to z. Therefore it can represent degenerate distributions and produce sharp samples.

However, GANs are difficult to train. One common failure is *mode collapse*, which means only few modes of samples are generated. Another problem is *diminished gradient*. When the discriminator gets too successful, the gradient

vanishes and the generator learns nothing. GAN is a kind of zero-sum non-cooperative game. It converges when the players reach a Nash equilibrium. In some cases, the loss may not converge using gradient descent in a minimax game, and the model parameters are oscillating with big swings. Various models were proposed to improve GANs. There are two branches: optimizing the loss function (e.g., unrolled GAN [55], Least-squared GAN [56], WGAN-GP [57]) and optimizing the architecture (e.g., DCGAN [58], InfoGAN [59], VAE-GAN [60]). Moreover, many tricks were involved to stabilize the training process, for example, using gradient penalty, batch normalization, and separated learning rates.

Another problem is the lack of a universal evaluation metric. A good metric should measure sample quality and diversity, check mode collapse, and detect over-fitting. Since the real data distribution is never unveiled, many researchers perform subjective human perceptual studies. Quantitative metrics such as Inception Score (IS) [61], Mode Score (MS) [62], Fréchet Inception Distance (FID) [63], and Kernel Inception Distance (KID) [64] are widely used to evaluate the diversity of generated samples. But people still lack a meaningful metric to guide the training process. Though many objective functions [65, 66] were proposed to measure the divergence between real and fake data, recent research [54] shows GANs can approach Nash equilibria via trajectories that do not necessarily reduce a divergence at each step.

GANs are the most successful generative models in the recent years. They can produce sharp, visually appealing, and high-resolution images. Many variants of GANs have been proposed in a wide range of applications including image generation [67, 68], super-resolution [69], image-to-image translation [70, 71], video synthesis [72], and semi-supervised learning [61].

2.2.3 Autoregressive Networks

Autoregressive networks are generative models for sequential data. Given a T-dimensional sequence $\mathbf{x} = (x_1, \cdots, x_T)^T$, we can factorize the joint distribution as a product of conditional probabilities over T time steps

$$p(\mathbf{x}) = \prod_{t=1}^{T} p(x_t \mid x_1, \cdots, x_{t-1}) \tag{2.17}$$

The chain rule factorization can be seen as a Bayesian network (Figure 2.4 top left). Originally, *autoregression* is a time series model that uses observations from previous time steps to predict the value at the current time step. A Bayesian network that does not rule out conditional dependence is said to have *autoregressive property*. Deep autoregressive networks (Figure 2.4 right) model conditional distributions by a stack of convolutional layers, where the output at time $t+1$ depends on its past values $x_t, x_{t-1}, \cdots$. The model does not output a value for x_{t+1}, but the predictive probability distribution $p(x_{t+1}|x_t, x_{t-1}, \cdots)$. Unlike the recurrent neural network (RNN), it is a feedforward network in which the past observations are not propagated via the hidden state, but directly provided by another input vector. Autoregressive networks are much faster than RNNs because they do not have recurrent connections.

DARN [73] is an early deep autoregressive network. It is a VAE with two hidden layers. The decoder includes autoregressive stochastic hidden units to capture high-level data structure and generate high-quality samples. Although autoregressive models are designed for sequential data, they can also be applied to produce non-sequential data. PixelCNN [74] models the joint distribution of pixels over an image as in equation 2.17, where x_t denotes a single pixel. Every

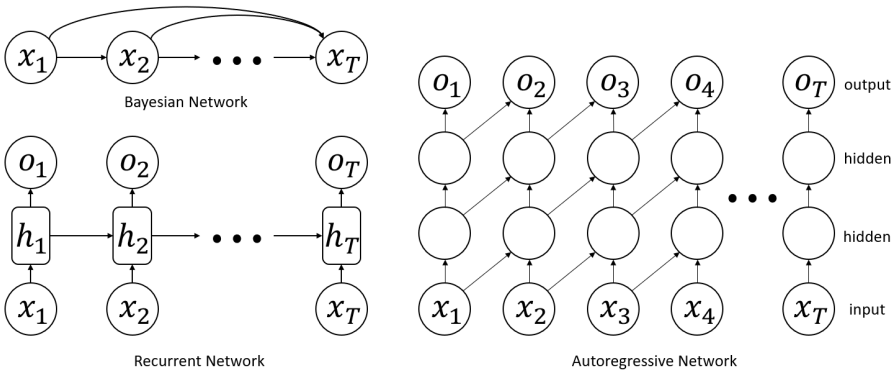

Figure 2.4: Comparing architectures of RNN and autoregressive network.

pixel depends on the previous pixels located above and to the left of it. The autoregressive networks are very flexible to model various kinds of data, like audio (WaveNet [75]), image(PixelCNN++ [76]), and text (Transformer [77]).

GANs, VAEs and autoregressive networks are the three most popular deep generative models. By improving the architecture, objective function, and training procedure, one hopes to generate more realistic images, audio, video and text. In addition, generative models are very amenable to conditioning. It can condition on random noise to generate new data, condition on data labels to help semi-supervised learning, condition on past data to predict time series, and condition on latent embeddings to learn disentangled representations. This thesis will investigate generative modeling in the perspective of game theory, and show some user cases in predictive modeling and unsupervised learning.

Chapter 3

Divergence Concepts

In generative modeling, it is a crucial issue to measure the similarity between real and synthetic data. This section will investigate several commonly used metrics to quantify the divergence between two probability distributions, when we only have access to samples drawn from them. Traditional approaches assume the real data distribution to be a density P_r, and parameterize the model distribution as one of the densities P_θ, $\theta \in \Theta$. The model is trained by maximizing the likelihood on real samples $x_i \in \mathcal{X}$.

$$\max_\theta \sum_i \log P_\theta(x_i) \tag{3.1}$$

This is equivalent to minimizing the Kullback-Leibler (KL) divergence between the real and fake data distributions $\mathbb{P}_r$, $\mathbb{P}_g$

$$D_{KL}(\mathbb{P}_r \parallel \mathbb{P}_g) = \int_{\mathcal{X}} P_r(x) \log \frac{P_r(x)}{P_g(x)} dx \tag{3.2}$$

Optimizing on this metric does not require to know the real distribution $P_r(x)$, and it has a unique minimum at $\mathbb{P}_r = \mathbb{P}_g$. However, KL divergence is not symmetric,

i.e., $D_{KL}(\mathbb{P}_r \parallel \mathbb{P}_g) \neq D_{KL}(\mathbb{P}_g \parallel \mathbb{P}_r)$. This property leads to unbalanced training outcomes. As shown in Figure 3.1, if $P_r(x) > P_g(x)$ then $\lim_{P_g(x)\to 0} D_{KL}(\mathbb{P}_r \parallel \mathbb{P}_g) = \infty$, and the loss is extremely high when the model distribution cannot cover the data (i.e., the phenomenon of "mode dropping"). In contrast, if $P_r(x) < P_g(x)$ then $\lim_{P_r(x)\to 0} D_{KL}(\mathbb{P}_r \parallel \mathbb{P}_g) = 0$, which means the KL does not punish at all when the model distribution drifts away from the data manifold.

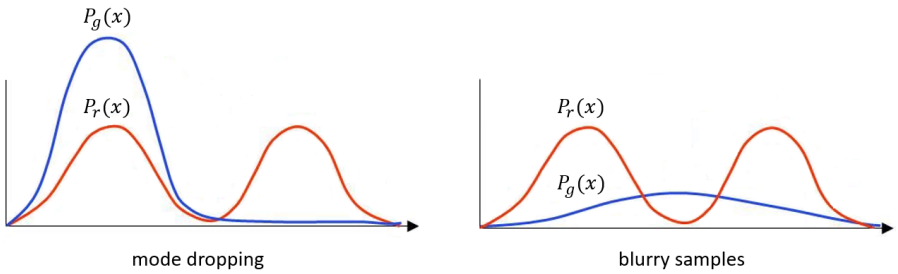

Figure 3.1: KL divergence is high in the left, but low in the right.

Moreover, a well defined KL divergence assumes $support(P_r) \subseteq support(P_g)$, that is, $P_g(x)$ cannot be zero in the region where $P_r(x) > 0$. This is not a trivial problem when the two manifolds are not perfectly aligned, and their intersections do not have full dimension. Therefore many traditional generative models include a high bandwidth noise component to cover all training samples. As a consequence, the models can avoid mode dropping but tend to generate non-realistic samples. This is consistent with the empirical finding that VAEs often produce blurry images, since the objective minimizes a KL divergence.

Minimax GANs [53] have been shown to minimize a Jensen-shannon divergence (JSD) when the critic is completely optimized in each generator's step. The JSD is defined as the average of two symmetric KL divergence.

$$JSD(\mathbb{P}_r \parallel \mathbb{P}_g) = D_{KL}(\mathbb{P}_r \parallel \mathbb{P}_m) + D_{KL}(\mathbb{P}_g \parallel \mathbb{P}_m) \tag{3.3}$$

where $\mathbb{P}_m = \frac{1}{2}(\mathbb{P}_r + \mathbb{P}_g)$ is a density defined on the intersection of $support(P_r)$ and $support(P_g)$. Therefore JSD is a symmetric metric and has support everywhere. It assigns high cost for samples not lie in the data manifold. GAN models are trained by playing a game in which JSD ensures there is a unique Nash equilibrium $\mathbb{P}_r = \mathbb{P}_g$, and punishes samples with high probability of being fake.

3.1 Bregman divergence

Bregman divergence is a generalized measure of distance between two points. Let f be a strictly convex and differentiable function $f : \mathcal{X} \to \mathbb{R}$. Given two points $x, y \in \mathcal{X}$, the Bregman divergence $D_f^{BR}(x, y)$ is defined as the difference between $f(x)$ and the first-order Taylor expansion of f around y evaluated at x

$$D_f^{BR}(x, y) := f(x) - f(y) - f'(y)(x - y) \tag{3.4}$$

Bregman divergence unifies a rich class of commonly used loss functions. The most popular one is the squared Euclidean distance

$$D_f^{BR}(x, y) = ||x - y||^2,$$

where the generator function is $f = ||x||^2$. KL divergence is also a Bregman divergence, when f is the negative entropy function $f(x) = x \log x$. The convexity of f ensures that the Bregman divergence always satisfy the non-negativity $D_f^{BR}(x, y) \geq 0$ and $D_f^{BR}(x, x) = 0$. But it is not a true distance because the triangle inequality and symmetry are not guaranteed.

Bregman divergences are widely used in learning optimal models. The points x, y

can be generalized to probability distributions. Given an unknown data distribution P_r and a set of observations sampled from it, we need to learn a parameterized model P_θ that best matches the data. Since any positive linear combination of Bregman divergences remains a Bregman divergence, the problem can be solved by convex optimization $\min_\theta D_f^{BR}(P_\theta, P_r)$.

3.2 f-divergence

GANs can be extended to a more general variational divergence estimation approach [62] by replacing the objective function with arbitrary $f-$divergences. $f-$ divergence involves a class of metrics to measure the discrepancy between two probability distributions. Next, we introduce the notion of $f-$divergence and its properties.

Definition 3.1. Let m and $\tilde{m}$ be two probability measures over a space Ω such that $\tilde{m}$ is absolutely continuous with respect to m. Then, for a convex function f such that $f(1) = 0$, the $f-$divergence of $\tilde{m}$ from m is defined as

$$D_f(m \parallel \tilde{m}) \equiv \int_\Omega f\left(\frac{dm}{d\tilde{m}}\right) d\tilde{m},$$

where $\frac{dm}{d\tilde{m}}$ is the Radon-Nikodym derivative of measure m w.r.t. measure $\tilde{m}$.

By Jensen's inequality,

$$\begin{aligned} D_f(m \parallel \tilde{m}) &= \int_\Omega f\left(\frac{dm}{d\tilde{m}}\right) d\tilde{m} \\ &\geq f\left(\int_\Omega \frac{dm}{d\tilde{m}}\, d\tilde{m}\right) \\ &= f\left(\int_\Omega d\tilde{m}\right) = f(1) = 0 \end{aligned} \tag{3.5}$$

Thus, $D_f(m \parallel \tilde{m})$ is non-negative for any convex function f. But $f-$divergence is not a distance, because it does not satisfy the symmetry property.

For any convex and lower-semicontinuous function f, the Legendre-Fenchel duality holds: $(f^*)^* = f$. f^* is the convex conjugate of f (also referred to as the Legendre transform), and it is also convex and lower-semicontinuous.

$$\begin{aligned} f^*(s) &= \sup_x [\langle x, s\rangle - f(x)] = [sx - f(x)] \mid_{x=(f')^{-1}(s)} \\ f(x) &= \sup_s [\langle x, s\rangle - f^*(s)] = [sx - f^*(s)] \mid_{s=((f^*)')^{-1}(x)} \end{aligned} \tag{3.6}$$

where $\langle x, s\rangle$ denotes the dot product of x and s, and f' is the first order derivative of f. As shown in Figure 3.2, the conjugate variable s can be viewed as the slope of the tangent line to the graph of f, and $f^*(s)$ is the negative $y-$intercept of the tangent line. The Legendre transformation builds a dual relationship between points and lines.

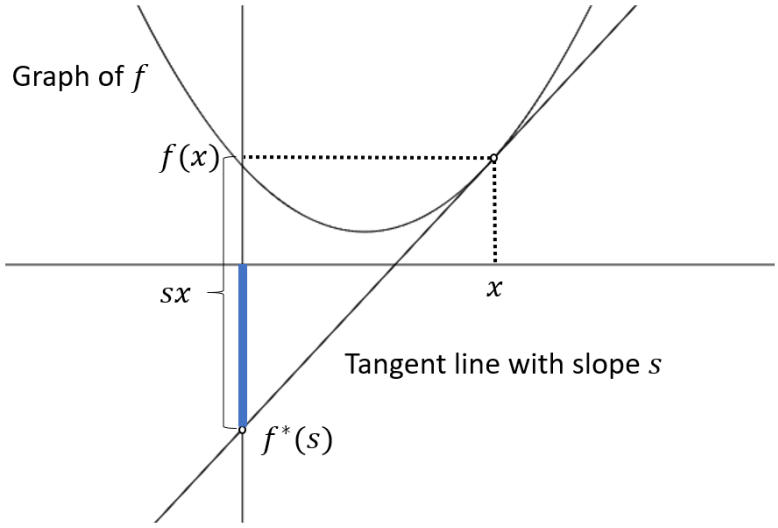

Figure 3.2: One-dimensional illustration of Legendre transform

The $f-$divergence is non-negative and has a lower bound. Let $P_r(x) \sim m$,

$P_g(x) \sim \tilde{m}$, $\xi(x) \in \Xi$, and Ξ is a subset of all possible functions $\Xi : \mathcal{X} \to \mathbb{R}$. By Legendre transform and Jensen's inequality,

$$\begin{aligned} D_f(\mathbb{P}_r \parallel \mathbb{P}_g) &= \int_{\mathcal{X}} P_g(x) f\left(\frac{P_r(x)}{P_g(x)}\right) dx \\ &= \int_{\mathcal{X}} P_g(x) \sup_s [\langle \frac{P_r(x)}{P_g(x)}, s\rangle - f^*(s)] dx \\ &\geq \sup_{\xi(x)\in\Xi} [\int_{\mathcal{X}} P_r(x)\xi(x)dx - \int_{\mathcal{X}} P_g(x) f^*(\xi(x))dx] \\ &= \sup_{\xi(x)\in\Xi} (\mathbb{E}_{x\in\mathbb{P}_r}[\xi(x)] - \mathbb{E}_{x\in\mathbb{P}_g}[f^*(\xi(x))]) \end{aligned} \tag{3.7}$$

This bound is tight when f is differentiable and the conjugate variable is set as

$$s = \xi(x) = f'\left(\frac{P_r(x)}{P_g(x)}\right) \tag{3.8}$$

For example, KL divergence is a special case of $f-$divergence when $f(x) = x\log x$; and minimax GANs correspond to the case when $f(x) = x\log x - (x+1)\log(x+1)$.

Generative modeling typically discusses three types of problems:

- **Modeling** Given a set of real samples $\{x_1, \cdots, x_n\} \in \mathcal{X}$, learn the unknown data distribution $P_r(x)$ by a parameterized model $P_\theta(x)$.
- **Sampling** Draw synthetic samples from the model distribution $\mathbb{P}_\theta$.
- **Evaluation** Given a sample x_i, evaluate the likelihood of being real.

Theoretically, given sufficient training samples and enough model capacity, the true distribution can be approximated well such that $\mathbb{P}_\theta$ converges to $\mathbb{P}_r$. But in practice, training GANs are often unstable. It is partly due to the fact that $\theta \mapsto D_f(\mathbb{P}_r \parallel \mathbb{P}_\theta)$ is not a continuous mapping. Next section will introduce a weaker distance metric to make it easier for the distributions to converge.

3.3 Wasserstein Distance

Wasserstein distance is a widely used metric to compare distributions. It measures the discrepancy between two probability distributions by estimating the *optimal transport cost*.

3.3.1 Optimal Transportation

Suppose we have two probability spaces $(\mathcal{X}, \mu)$ and $(\mathcal{Y}, \nu)$, the optimal transport cost between the two measures is

$$C(\mu, \nu) = \inf_{\pi \in \Pi(\mu, \nu)} c(x, y) d\pi(x, y), \tag{3.9}$$

where $c(x, y)$ is the cost of moving one unit of mass from point x to point y, and $\Pi(\mu, \nu)$ is the set of all joint probability measures on $\mathcal{X} \times \mathcal{Y}$ such that μ and ν are marginals on $\mathcal{X}$ and $\mathcal{Y}$ respectively. The joint measures are called transport plans [78], and Equation 3.9 is to estimate the minimum cost among all transport plans. The problem is illustrated in Figure 3.3. Imagine we have a sand pile, and hope to transport it to another place for construction. The initial and final shapes are known, but the transport plan needs to be determined: to which location should each sand particle be moved to? The transport plan offers a recipe that associates

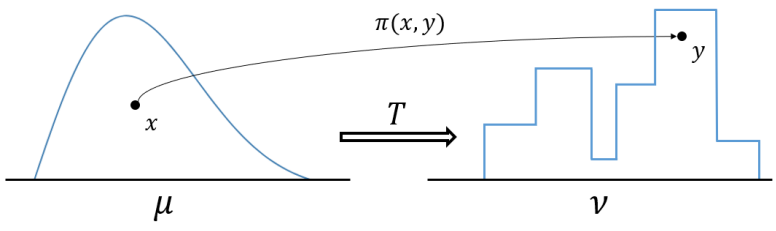

Figure 3.3: An illustration of optimal transportation

each point x with a destination y. It can be represented by a function $T : \mathcal{X} \to \mathcal{Y}$ that changes the mass from μ to ν. For all $\nu-$integrable functions ϕ, we have

$$\int_{\mathcal{Y}} \phi(y) d\nu(y) = \int_{\mathcal{X}} \phi(T(x)) d\mu(x) \tag{3.10}$$

There are infinite number of transport plans between two given probability measures. An optimal transport is the plan that transport the mass at minimum cost. The transport cost of a certain particle is defined by the product of its mass by the distance being moved. During transportation, the initial distribution changes smoothly to the final one, and generates a sequence of distributions in between, known as the displacement interpolation [79]. Sometimes, people do not care about the optimal transport plan, but only need the minimum cost value. But investigating the displacement interpolations is helpful to find an optimal learning path in geometric domains, and uncover the intermediate states between two probability measures.

3.3.2 Definition and Properties

Wasserstein distance is a specific kind of optimal transport cost in which the ground cost $c(x, y) = d(x, y)$ is a distance function. The p^{th} Wasserstein distance ($p \geq 1$) is defined as follows

Definition 3.2. Let $(\mathcal{X}, d)$ be a complectly separable metric space, μ and ν be two probability measures on $\mathcal{X}$. For $p \geq 1$, the $p-$Wasserstein distance is

$$W_p(\mu, \nu) = \left(\inf_{\pi \in \Pi(\mu,\nu)} \int_{\mathcal{X}\times\mathcal{X}} d(x, y)^p d\pi(x, y) \right)^{\frac{1}{p}} \tag{3.11}$$

With the same notations, the Wasserstein space of order p is defined as the collection of all probability measures μ on $\mathcal{X}$

$$P_p(\mathcal{X}) := \left\{ \mu \in P(\mathcal{X}) : \int_{\mathcal{X}} d(x_0, x)^p \mu(dx) < +\infty \right\} \tag{3.12}$$

Specifically, the $1-$Wasserstein distance W_1 is known as the earth mover's distance (EMD). It is widely used to compare discrete distributions such as images and histograms. When $\mu = \delta_x$ and $\nu = \delta_y$ are two Dirac distributions, there is only one feasible solution for the transportation problem. The $p-$Wasserstein distance degenerates to the ground cost of transporting the point mass from x to y.

$$W_p(\delta_x, \delta_y) = d^p(x, y)$$

The Wasserstein distance has many properties. W_p is a true distance because it is symmetric ($W_p(\mu, \nu) = W_p(\nu, \mu)$), non-negative ($W_p(\mu, \nu) \geq 0$ and $W_p(\mu, \nu) = 0$ if and only if $\mu = \nu$), and satisfies the triangle inequality.

$$\forall \mu, \nu, \omega \in \mathcal{X}, \qquad W_p(\mu, \nu) \leq W_p(\mu, \omega) + W_p(\omega, \nu) \tag{3.13}$$

More important, W_p is a weak distance. It allows quantitative comparison between two distributions with low-dimensional supports that do not overlap. In contrast, many traditional metrics are not defined on discrete space. For example, KL divergence requires the measure to be continuous and has a density w.r.t. a base measure. In addition, the Wasserstein distance has weak convergence in the Wasserstein space, that is, if μ is the target measure and $(\mu_k)_{k \in \mathbb{N}}$ is a sequence of

measures in $P_p(\mathcal{X})$, and $\rightarrow$ denotes weakly converge, then

$$\mu_k \rightarrow \mu \quad \Longleftrightarrow \quad W_k(\mu_k, \mu) \rightarrow 0$$

Based on the above properties, one can derive that Wasserstein distance W_p is continuous on $P_p(\mathcal{X})$. If $\mu_k \rightarrow \mu$, $\nu_k \rightarrow \nu$ as $k \rightarrow \infty$, then

$$W_p(\mu_k, \nu_k) \rightarrow W_p(\mu, \nu)$$

The weak topology and continuous property makes Wasserstein distance very tempting as an objective function, since the loss can change smoothly w.r.t. the model parameters. Thus the effective gradients are always available during training, and the convergence is ensured.

Information based metrics like $f-$divergence and Bregman divergence are commonly used as classification loss or fitting error in parameter estimation problems. They compare two probability measures by integrating the pointwise ratios between them, and have nice computational properties. But the entropy-based objective functions are not continuous and differentiable everywhere. For example, the KL divergence between two univariate Gaussians $\mathcal{N}(\mu_a, \sigma_a^2)$, $\mathcal{N}(\mu_b, \sigma_b^2)$ are

$$D_{KL} = \frac{1}{2}\left(\frac{\sigma_a^2}{\sigma_b^2} + \log\left(\frac{\sigma_b^2}{\sigma_a^2}\right) + \frac{|\mu_a - \mu_b|}{\sigma_b^2} - 1\right)$$

when σ_b diminishes and the Gaussian degenerates to a Dirac distribution, it will be infinitely far away from all other Gaussians in the KL geometry.

The Wasserstein metric introduces a distance over probability distributions by exploring the optimal transport geometry. It leverages the physical idea of mass

transportation to compare measures. The information divergences compare two measures by comparing their mass pointwise, and do not recognize the spatial relationship between them. For instance, $D_{KL}(m \parallel \tilde{m}) = \int_{\mathcal{X}} \log(\frac{m(x)}{\tilde{m}(x)}) m(x)dx$ is invariant of reversible transformations on $x = (x_1, x_2, \cdots)^T$ because the product $m(x)dx$ removes the dimensional information. This property is illustrated in Figure 3.4.

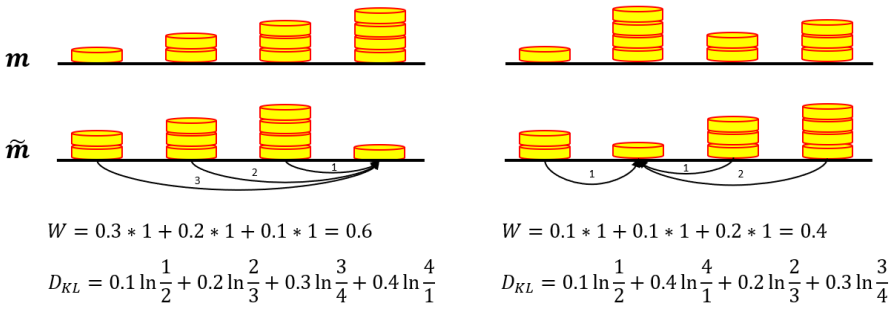

Figure 3.4: Wasserstein metric recognizes permutation changes, while KL divergence is permutation invariant as it calculates pointwise ratio between measures.

3.3.3 Approximating the Wasserstein Distance

In practice, the main challenge is to efficiently approximate and differentiate the Wasserstein distance. Recall that W_p is defined by a minimization problem on the transport cost between two probability measures (also known as the Monge–Kantorovich problem [78])

$$\inf_{\pi(\mu,\nu)} \int_{\mathcal{X}\times\mathcal{Y}} c(x,y)d\pi(x,y) \tag{3.14}$$

where the infimum is over all joint measures π on $\mathcal{X} \times \mathcal{Y}$ with marginals μ and ν. There could be infinite number of transport plans, i.e. the couplings of (μ, ν) is infinite, which makes the primal problem intractable.

The Monge–Kantorovich problem has a dual form

$$\sup_{\phi(y)-\psi(x)\leq c(x,y)} \left(\int_{\mathcal{Y}} \phi(y) d\nu(y) - \int_{\mathcal{X}} \psi(x) d\mu(x) \right) \tag{3.15}$$

where $\psi(x) : \mathcal{X} \to \mathbb{R}$ and $\phi(y) : \mathcal{Y} \to \mathbb{R}$ are integrable functions. The supremum in (3.15) is no more than the infimum in (3.14), and the bound is tight when

$$\begin{aligned} \phi(y) &= \inf_{x \in \mathcal{X}} (\psi(x) + c(x, y)) \\ \psi(x) &= \sup_{y \in \mathcal{Y}} (\phi(y) - c(x, y)) \end{aligned} \tag{3.16}$$

Thus the optimal transport cost can be interpreted as a maximization problem

$$\min_{\pi(\mu,\nu)} \int_{\mathcal{X}\times\mathcal{Y}} c(x, y) d\pi(x, y) = \sup_{\psi} \left(\int_{\mathcal{Y}} \psi^c(y) d\nu(y) - \int_{\mathcal{X}} \psi(x) d\mu(x) \right) \tag{3.17}$$

where $\psi^c(y) = \inf_{x\in\mathcal{X}}(\psi(x) + c(x, y))$ is a generalization of the Legendre transform.

Remark 3.3.1. (Special case for $p = 1$)
When $c(x, y) = d(x, y)$ is a distance on a Polish space $\mathcal{X}$ and μ, ν are functions integrable with their first moments, we have the Kantorovich's dual of the $1-$Wasserstein distance

$$W_1(\mu, \nu) = \sup_{||\nabla\psi||_1 \leq 1} \left(\int_{\mathcal{X}} \psi d\mu - \int_{\mathcal{X}} \psi d\nu \right) = \sup_{\psi \in Lip^1} \mathbb{E}_{x\sim\mu}[\psi(x)] - \mathbb{E}_{y\sim\nu}[\psi(y)] \tag{3.18}$$

in which the supremum runs over all $1-$Lipschitz functions ψ. It becomes $k \cdot W_1$ if we change the constraint to $k-$Lipschitz functions $||\nabla\psi||_1 \leq k$.

Although Wasserstein spaces are not Hilbertian in general, it is the case for

univariate distributions. It means the 1D Wasserstein distance can be efficiently approximated by a Hilbertian metric on some feature representations of the probability measures.

Remark 3.3.2. (Special case for 1D Wasserstein distance)
When $\mathcal{X} = \mathbb{R}$, let F_μ and F_ν be the cumulative distribution functions (CDF) for probability distributions μ and ν. In the 1D case, there exist inverse CDFs F_μ^{-1} and F_ν^{-1}. Then the Wasserstein distance can be computed by the integration of their inverse CDFs

$$W_p(\mu, \nu) = \int_0^1 |F_\mu^{-1}(x) - F_\nu^{-1}(x)|^p dx \tag{3.19}$$

In particular, the formula becomes even simpler for $p = 1$

$$W_1(\mu, \nu) = \int_{\mathbb{R}} |F(x) - G(x)| dx = \int_{\mathbb{R}} \left| \int_{-\infty}^{x} d(\mu(x) - \nu(x)) \right| dx \tag{3.20}$$

3.3.4 Displacement Interpolation: A Case Study

Time-dependent mass transportation leads to continuous displacement of probability measures. In generative modeling, the learning process pushes the model distribution P_g toward the data distribution P_r. Suppose the model is initialized as $P_g = f_0$, and updated to f_t at time step t. A successful algorithm should generate a sequence of distributions $f_0, f_1, \cdots, f_T$ that converge to the target distribution P_r. The interpolations form a path in the metric space. Thus the optimal transport from the initial measure to the final measure is obtained by minimizing the total energy spent on each path.

The following example shows that Wasserstein metric prefers displacement

interpolation than linear interpolation, which means it can keep modes (e.g., the Gaussian peak) and reflect translational motions between two objects. Figure 3.5 shows that under Wasserstein metric, displacement interpolation (up) has lower cost than linear interpolation (down), while KL-divergence shows the opposite.

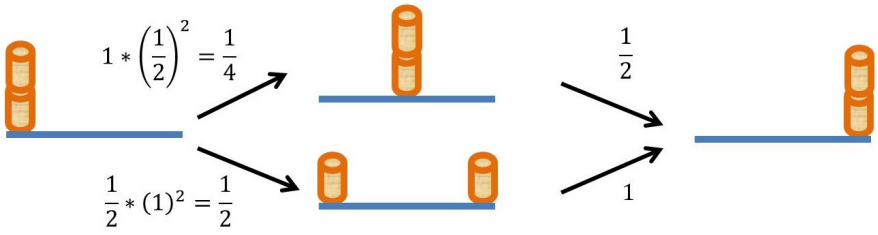

Figure 3.5: Wasserstein metric prefers displacement interpolation to linear interpolation. Up: $W_2(t_0, t_1) + W_2(t_1, t_2) = \frac{1}{4} + \frac{1}{4} = \frac{1}{2}$. Down: $W_2(t_0, t_1) + W_2(t_1, t_2) = \frac{1}{2} + \frac{1}{2} = 1$. The KL will never choose the upper path because the cost is infinite.

A Toy Example This example shows learning the optimal transportation between two density distributions. The initial and target distributions are given by two Gaussians (see Figure 3.6).

$$\begin{aligned} f_0 &= \mathcal{N}(\mu = (0.2, 0.3), \sigma = 0.1) \\ f_1 &= \mathcal{N}(\mu = (0.6, 0.7), \sigma = 0.07) \end{aligned} \tag{3.21}$$

Initially, the intermediate states are set as linear interpolations, where a unimodal distribution is changed to bimodal. This is not desirable because it doesn't keep the mode. (think of an object being split into two parts while moving and then put

Figure 3.6: Initial and target distributions f_0, f_1.

together, as the lower part of Figure 3.5). It has been proved in [80] that optimizing with Wasserstein loss can preserve variance of the intermediate distributions, thus the object moves from one position to another without changing its shape. Figure 3.7 shows that after 1000 iterations, the Gaussian peak is preserved in the intermediate distributions, and the loss decreases to the optimal transport cost (see Figure 3.8).

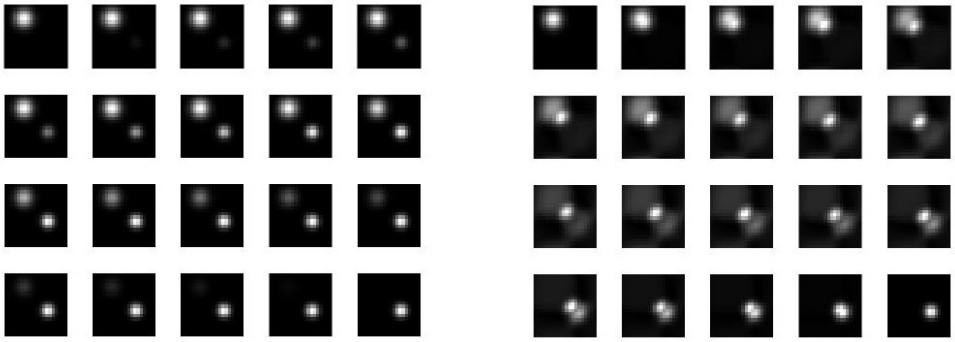

Figure 3.7: Learn the optimal transportation between two Gaussians. Left: Linear interpolation (iter 0). Right: Displacement interpolation (iter 1000)

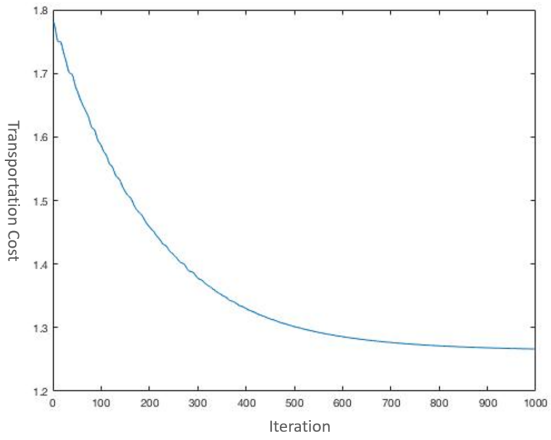

Figure 3.8: Transportation cost during learning.

Chapter 4

Distributionally Robust Games

In this chapter, we introduce the framework of distributionally robust games (DRG). These are multi-player games in which each player has to choose her action in an uncertain environment. The decision-maker has a continuous action space and aims to learn her optimal strategy. The true distribution of the uncertainty is unknown to the players. To deal with the uncertain environment, each player models the state of nature using a worst-case distribution, also called *adversarial distribution*. Thus each player's payoff depends on the other players' decisions and the decision of a virtual player (i.e., nature) who selects an adversarial distribution among all possible scenarios. We provide alternative ways to select the adversarial distribution based on empirical observations of the decision-maker. This leads to a distributionally robust optimization problem. The problem is formulated using a notion of divergence between two measures: the modified measure and the exact measure associated to the uncertainty. Simple algorithms, whose dynamics are inspired from gradient flows, are proposed to find local optima. The methodology does not require strong convexity assumptions as in the classical gradient algorithms.

Theoretical findings are illustrated in a convex setting and its limitations are tested with a non-convex non-concave function. The method is extended to a class of optimization problems and robust games. Illustrative examples and simulations are provided. This chapter is based on our book chapter "Distributionally Robust Optimization" [18] published in 2018. The main contributions are summarized as follows:

- We formulate the distributionally robust game by the statistical notions of $f-$divergence and Wasserstein distance between two distributions, here represented by the adversarial distribution and the exact distribution.
- We develop *triality theory* to reduce the complexity of the problem, and provide a computational method based on Bregman flow for approximately computing the robust equilibria.
- We introduce a class of distributionally robust games with continuous action spaces, for which a subset of equilibria can be computed using the Bregman algorithm.

4.1 Introduction

Games with payoff uncertainty refer to games in which the outcome of an action is not known with certainty by the players. Such games are also called incomplete information games and can be formalized in different ways. Distribution-free models of incomplete information games, both with and without private information are examined in [20, 81]. There the players use a robust optimization approach to contend with the worst-case scenario payoff. Robust optimization can be defined as

the process of determining the best or most effective response, utilizing a quantitative measurement system under worst-case uncertain functions or parameters. The distribution-free approaches relax the well-known *Bayesian game* model proposed by Harsanyi [32]. The limitation of the distribution-free model is that the uncertainty parameters have to be carefully defined, and the bounds given by the worst case scenario may not be useful in many interesting problems. In many cases such approach leads to too conservative and unrealistic scenarios.

We introduce a distributionally robust game with continuous action spaces. This game provides a new and original way of addressing game scenarios with incomplete information. The relevance in formulating such a new game is in that it relaxes the assumptions of Harsanyi's Bayesian games [32]. Instead of assuming an underlying mean-field or exactly known probability distribution, one acts with an uncertainty set, which could be distributions chosen by other players. In the distribution-free approach, the interval (or generally the uncertainty set) needs to be known or learnable by the decision-makers. In contrast to this, the distributionally robust approach may test any alternative distribution within a divergence ball defined by $f-$divergence or Wasserstein metric. The distributional robustness method works on probability distributions instead of uncertainty parameters or functions. The worst-case distribution within a certain carefully designed distributional uncertainty set may provide interesting features. The set of distributions should be chosen to fit for the applications at hand.

In robust best-response problems, the uncertain sets are represented by deterministic models. The opponent players have a bounded capability to change the uncertain parameters, and therefore affects the objective function the decision maker seeks to optimize. Each player has her own robust best-response optimiza-

tion problem to solve. Thus, the standard best-response problem of player j: $\inf_{a_j \in A_j} l_j(a_j, a_{-j}, \omega)$ becomes the minimax robust best response

$$\inf_{a_j \in A_j} \sup_{\omega \in \Omega} l_j(a_j, a_{-j}, \omega) \tag{4.1}$$

where a_j is the action to choose and ω is the objective functional evaluated at uncertain state ω. This approach to uncertainty has a long history in optimization, control and game theory [82, 83, 84].

A credible alternative to this set-based uncertainty is to use a stochastic model, in which the uncertain state ω is a random variable with distribution m. If we assume the generating mean-field distribution m is known, it becomes a standard stochastic optimal control paradigm. If m is not known and the only known is a set of distributions lie in the neighborhood of m: $m' \in B_\rho(m)$, the resulting best response to mean-field formulation is the so-called distributionally robust best response

$$\inf_{a_j \in A_j} \sup_{m' \in B_\rho(m)} \mathbb{E}_{\omega \sim m'} l_j(a_j, a_{-j}, \omega) \tag{4.2}$$

Distributionally robust optimization can be used not only to provide a robust solution to a problem when the true distribution is unknown, but it also can, in many cases, give a general solution taking into account some risk. For instance, the best response of Bob in Example 2.5 is to choose the Nash equilibrium W, but the risk-aware solution will be a safer strategy U that guarantees his minimum payoff. We give a rigorous definition of distributionally robust equilibrium, and provide a novel class of risk-aware Bregman algorithms for global optimization and cooperation solution seeking problems.

The distributionally robust optimization problems involves naturally the space

of probability distributions which is of infinite dimensions for continuous action spaces. While there are several differentiability concepts on the space of probability measures (Fréchet, Gâteaux, Wasserstein-gradient, etc), their computation is not without challenges. We develop a triality theory to reformulate the problem and reduce considerably the curse of dimensionality. We provide a computational method based on Bregman flow for approximately computing equilibria. Such a computational method allows us to test the implementability of the approach on numerical data, and illustrative examples are given. We show that the resulting iterative dynamics, which we call Bregman dynamics, is characterized by double (or higher) exponential decay and converges to distributionally robust equilibria.

4.2 Problem Formulation

This section introduces distributionally robust game models. We will first introduce the concept of distribution uncertainty set, and then present a generic formulation of the problem. After that, we will discuss triality theory and apply such theory to reduce the model complexity via Lagrangian relaxation. Individual components of the optimization and their solvability issues via equivalent formulations will be discussed.

4.2.1 Distribution Uncertainty Set

Consider a decision-maker who wants to select an action $a \in \mathcal{A}$ in order to optimize her objective $l(a, \omega)$, where ω is an uncertain state. The information structure is the following

- The true distribution of ω is not known to the decision-maker, neither is the

upper/lower bound of ω.

- A decision-maker can measure/observe realization of the random variable ω.

The decision-maker chooses to experiment several trials and obtains statistical realizations of ω from measurements. The measurement data can be noisy, imperfect and erroneous. Then, an empirical distribution (or histogram) m is built from the realizations of ω. However, it is not the true distribution of the random variable ω, and it may not be a reliable measure due to statistical, bias, measurement, observation or computational errors. Therefore, the decision maker is facing a risk. The risk-sensitive decision-maker should decide actions that improve the expected payoff $\mathbb{E}_{m'} l(a, \omega)$ among alternative distributions m' within a certain level of deviation ρ from the distribution m. All alternative admissible distributions are defined in an uncertainty set.

Let $(\Omega, \mathcal{F}, m)$ be a probability space. Here m is a probability measure defined on $(\Omega, \mathcal{F})$. The distribution m of the state ω is used to capture the probability of the different scenarios and of the corresponding payoff function obtained under each of such scenarios for fixed action profile. We assume that the exact distribution of the state is not available in general. Then we propose an uncertainty/constraint set among all the possible distributions with a divergence bounded above by a scalar value ρ. Such a constraint takes different forms that depend on the distance metric. We present two kinds of uncertainty sets defined on $f-$divergence and Wasserstein metric.

$$B_\rho^f(m) = \left\{ m' \mid \int_\Omega dm' = m'(\Omega) = 1, D_f(m' \parallel m) \leq \rho \right\} \tag{4.3}$$

$$B_\rho^W(m) = \left\{ m' \mid \int_\Omega dm' = m'(\Omega) = 1, W_p(m', m) \leq \rho \right\} \tag{4.4}$$

Recall that for $f-$divergence m' needs to be absolutely continuous w.r.t. m.

4.2.2 Definition

In distributionally robust games, each player j chooses $a_j \in \mathcal{A}_j$ to optimize the worst loss functional $\mathbb{E}_{m'}(a_j, \omega)$ subject to the constraint that $D_f(m' \parallel m) \leq \rho$ or $W_p(m', m) \leq \rho$. This means the worst loss performance is obtained under the assumption that a virtual player (nature) acts as acts as a discriminator who pushes the distribution m' away from m with an effort capacity that should not exceed $\rho > 0$. The robust stochastic optimization of player j given $(a_{j'})_{j' \neq j}$, m, ρ is

$$(\mathcal{P}_j) : \inf_{a_j \in \mathcal{A}_j} \sup_{m' \in B_\rho(m)} \mathbb{E}_{\omega \sim m'} l_j(a, \omega) \tag{4.5}$$

The alternative distribution m' is not a given profile, and it could be deformed or falsified by the discriminator. We assume that: for uncertainty set defined on $f-$divergence the measure m' is continuous w.r.t. m. The loss function $l_j(\cdot, \omega)$ is proper and upper semi-continuous for $m-$almost all $\omega \in \Omega$. Either the domain $\mathcal{A}_j$ is a non-empty compact set or $\mathbb{E}_{m'} l_j(a, \omega)$ is coercive. Thus we have the definition

Definition 4.1. *(Distributionally Robust Game)*
The distributionally robust game $\mathcal{G}(m)$ involves

- the set of players $\mathcal{J} = \{1, 2, \ldots, n\},\ n \geq 2$
- the decision space $\mathcal{A}_j$ of each player j, $j \in \mathcal{J}$
- the uncertainty set $B_\rho(m)$ defined on Ω and ρ
- the payoff function $\mathbb{E}_{m'} l_j(a, \omega)$ of player j, $j \in \mathcal{J}$.

With the above game in mind, we can introduce the following solution concept.

Definition 4.2. *(Distributionally Robust Equilibrium)*
Let a_j^* be the configuration of player j and $a_{-j}^* := (a_k^*)_{k\neq j}$. A strategy profile $a^* = (a_1^*, \ldots, a_n^*)$ satisfying

$$\sup_{m'\in B_\rho(m)} \mathbb{E}_{m'} l_j(a^*, \omega) \leq \sup_{m'\in B_\rho(m)} \mathbb{E}_{m'} l_j(a_j, a_{-j}^*, \omega), \tag{4.6}$$

for every $a_j \in \mathcal{A}_j$ and every agent j, is said *distributionally robust pure Nash equilibrium* of the game $\mathcal{G}(m)$.

We have defined the game and its solution concept. The task now is to solve problem (4.5) under both $f-$divergence and Wasserstein metric. One of the difficulties of the problem is the curse of dimensionality. The distributionally robust optimization problem (4.5) of the decision-maker is an infinite dimensional robust optimization problem because B_ρ is of infinite dimensions. Below we will show that problem (4.5) can be transformed into an optimization in the form of sup inf sup, which has three alternating terms. Solving this problem requires a triality theory.

4.2.3 Triality Theory

We here present the duality gap and develop a triality theory. Consider uncoupled domains $A_j, j \in \{1, 2, 3\}$. For a general function l_2, one has

$$\sup_{a_2\in\mathcal{A}_2} \inf_{a_1\in\mathcal{A}_1} l_2(a_1, a_2) \leq \inf_{a_1\in\mathcal{A}_1} \sup_{a_2\in\mathcal{A}_2} l_2(a_1, a_2) \tag{4.7}$$

and the difference

$$\min_{a_1\in\mathcal{A}_1}\max_{a_2\in\mathcal{A}_2} l_2(a_1,a_2) - \max_{a_2\in\mathcal{A}_2}\min_{a_1\in\mathcal{A}_1} l_2(a_1,a_2) \tag{4.8}$$

is the duality gap. As it is widely known in duality theory from Sion's Theorem [83] (which is an extension of von Neumann's minimax Theorem) there is an equality, for example for convex-concave function, and the value is achieved by a saddle point in the case of non-empty convex compact domain.

Triality theory focuses on optimization problems of the forms $\sup\inf\sup$ or $\inf\sup\inf$. The term triality is used here because there are three key alternating terms in these optimizations.

Proposition 4.2.1. Let $(a_1,a_2,a_3)\mapsto l_3(a_1,a_2,a_3)\in\mathbb{R}$ be a function defined on the product space $\prod_{j=1}^{3}\mathcal{A}_j$. Then, the following inequalities hold

$$\begin{aligned}\sup_{a_2\in\mathcal{A}_2}\inf_{a_1\in\mathcal{A}_1,a_3\in\mathcal{A}_3} l_3(a_1,a_2,a_3) &\le \inf_{a_3\in\mathcal{A}_3}\sup_{a_2\in\mathcal{A}_2}\inf_{a_1\in\mathcal{A}_1} l_3(a_1,a_2,a_3)\\ &\le \inf_{a_1\in\mathcal{A}_1,a_3\in\mathcal{A}_3}\sup_{a_2\in\mathcal{A}_2} l_3(a_1,a_2,a_3)\end{aligned} \tag{4.9}$$

and similarly,

$$\begin{aligned}\sup_{a_1\in\mathcal{A}_1,a_3\in\mathcal{A}_3}\inf_{a_2\in\mathcal{A}_2} l_3(a_1,a_2,a_3) &\le \sup_{a_3\in\mathcal{A}_3}\inf_{a_2\in\mathcal{A}_2}\sup_{a_1\in\mathcal{A}_1} l_3(a_1,a_2,a_3)\\ &\le \inf_{a_2\in\mathcal{A}_2}\sup_{a_1\in\mathcal{A}_1,a_3\in\mathcal{A}_3} l_3(a_1,a_2,a_3)\end{aligned} \tag{4.10}$$

Proof. In Appendix A.1 □

4.2.4 Equivalent Formulations

Assume that $a \mapsto \mathbb{E}_m l_j(a,\omega)$ is continuous for $m-$almost all ω. Then, the functional $F_j : \ m \mapsto \inf_{a_j} \mathbb{E}_m l_j(a,\omega)$ is Gateaux differentiable with derivative

$$F_{j,m}(m') = \inf_{a_j \in \mathcal{A}_j^*(m)} \mathbb{E}_{m'} l_j(a,\omega),$$

where $\mathcal{A}_j^*(m) = \arg\min_{a_j} \mathbb{E}_m l_j(a,\omega)$ is the best-response under m. This derivative in the space of square integrable measurable functions under m which is of infinite dimensions, does not facilitate the computation of the robust optimal strategy a_j^*, m'^*. Below we explain how the dimensionality of problem (4.5) can be significantly reduced using a representation by means of the triality theory inequalities of Proposition 4.2.1. We propose equivalent formulations for both models defined by $f-$divergence and Wasserstein metric.

4.2.4.1 $f-$divergence

When $\mathcal{A}$ are of finite dimensions, the distributionally robust optimization problem (4.5) under $f-$divergence is equivalent to a finite dimensional stochastic optimization problem. To see this, the original problem need to be transformed. Let us introduce the likelihood functional $L(\omega') = \frac{dm'}{dm}(\omega')$, and set

$$L_\rho(m) = \left\{ L \mid \int_{\omega'} f(L(\omega'))dm - f(1) \leq \rho \right\}$$

Then, the robust best-response problem of agent j is equivalent to

$$\inf_{a_j \in \mathcal{A}_j} \sup_{L \in L_\rho(m)} \mathbb{E}_m[l_j L] \tag{4.11}$$

We introduce the Lagrangian as

$$\tilde{l_j}(a, L, \lambda, \mu) = \int_{\omega'} l_j(a, \omega')L(\omega')dm(\omega') - \lambda\left(\rho + f(1) - \int_{\omega'} f(L(\omega'))dm(\omega')\right) - \mu\left(1 - \int_{\omega'} L(\omega')dm(\omega')\right), \tag{4.12}$$

where $\lambda \geq 0$ and $\mu \in \mathbb{R}$. The problem solved by player j becomes

$$(\mathcal{P}_j^*) : \sup_{a_j \in \mathcal{A}_j} \inf_{L \in L_\rho(m)} \sup_{\lambda \geq 0, \mu \in \mathbb{R}} \tilde{l_j}(a, L, \lambda, \mu) \tag{4.13}$$

A full understanding of problem (4.13) requires the triality theory. The underlying idea is that one can use a transformation of the last two terms to derive a finite dimensional optimization. problem. The Lagrangian $\tilde{l_j}$ of player j is clearly concave in L and convex in λ, μ, and is semi-continuous jointly. By the triality theory $\tilde{l_j}$ satisfies the sup inf inequality and we have the following

$$\inf_{a_j \in \mathcal{A}_j} \sup_{L \in L_\rho(m)} \inf_{\lambda \geq 0, \mu \in \mathbb{R}} \tilde{l_j}(a, L, \lambda, \mu) \leq \inf_{a_j \in \mathcal{A}_j} \inf_{\lambda \geq 0, \mu \in \mathbb{R}} \sup_{L \in L_\rho(m)} \tilde{l_j}(a, L, \lambda, \mu)$$

The latter problem can be rewritten as

$$(\tilde{\mathcal{P}}_j^*) : \inf_{a_j \in \mathcal{A}_j, \lambda \geq 0, \mu \in \mathbb{R}} \sup_{L \in L_p(m)} \tilde{l_j}(a, L, \lambda, \mu) \tag{4.14}$$

The Lagrangian function takes the form as

$$\tilde{l_j} = \lambda(\rho + f(1)) + \mu + \int \{L[l_j - \mu] - \lambda f(L)\}dm \tag{4.15}$$

It follows that

$$\sup_{L\in L_\rho(m)} \tilde{l}_j(a,L,\lambda,\mu) = \lambda(\rho+f(1))+\mu+\sup_{L\in L_\rho(m)}\int\{L[l_j-\mu]-\lambda f(L)\}dm$$

Introduce the Fenchel-Legendre transform on L and exchanging sup and $\int$, one gets

$$\sup_{L\in L_\rho(m)} \tilde{l}_j(a,L,\lambda,\mu) = \lambda(\rho+f(1))+\mu+\int \lambda f^*(\frac{l_j-\mu}{\lambda})dm \tag{4.16}$$

where

$$f^*(\xi) = \sup_L[\langle L,\xi\rangle - f(L)] = -\inf_L[f(L)-\langle L,\xi\rangle].$$

Since $\mathcal{A}_j\times\mathbb{R}_+\times\mathbb{R}$ is a subset of a finite dimensional vector space (dimension $n+2$), it follows that the robust best-response problem of player j is equivalent to the finite dimensional stochastic optimization problem:

$$(\mathcal{P}_j^*)\begin{cases} \inf_{a_j\in\mathcal{A}_j,\lambda\geq 0,\mu\in\mathbb{R}} l_j^*(a,\lambda,\mu,m) \\ l_j^*(a,\lambda,\mu,m) = \lambda(\rho+f(1))+\mu+\int\lambda f^*(\frac{l_j-\mu}{\lambda})dm = \mathbb{E}_m[h_j]. \end{cases} \tag{4.17}$$

where h_j is the integrand cost $\lambda(\rho+f(1))+\mu+\int\lambda f^*(\frac{l_j-\mu}{\lambda})$.

We have converted the infinite dimensional problem (P_j) into a finite dimensional problem (P_j^*). The above calculations culminate in the following result:

Proposition 4.2.2. If $a,\ \lambda^*(a),\ \mu^*(a)$ is a solution of (P_j^*) then the optimal likelihood L^* is such that $\int_\omega L^*dm = 1$, $f'(L^*) = \frac{l_j-\mu^*}{\lambda}$. This means that a_j and $d\tilde{m}^* = L^*dm$ provide a solution of the original problem (P_j).

Proof. In Appendix A.2 □

4.2.4.2 Wasserstein metric

Similarly, when $\mathcal{A}_j$ is a set of finite dimension, the distributionally robust optimization problem under Wasserstein metric is equivalent to the finite dimensional stochastic optimization problem. If the function $\omega \mapsto l_j(a, \omega)$ is upper semi-continuous and (Ω, d) is a Polish space, then the distributionally robust optimization problem define by $W_p(m, m')$ is equivalent to

$$\begin{cases} \sup_{a_j \in \mathcal{A}_j} \inf_{m' \in B_\rho^W(m)} \mathbb{E}_{m'}[l_j] = \sup_{a_j \in \mathcal{A}_j} \sup_{\lambda \geq 0} \mathbb{E}_m[h_j] \\ h_j = \lambda\rho + \sup_{\omega' \in \Omega}[l_j(a, \omega') - \lambda d^p(\omega, \omega')] \end{cases} \tag{4.18}$$

It is a finite dimension problem on $\mathcal{A}_j \times \mathbb{R}_+ \times \Omega$ if $\mathcal{A}_j$ and Ω have finite dimensions. More details about distributionally robust games under Wasserstein metric will be discussed in Chapter 5.

4.2.4.3 Existence of distributionally robust Nash equilibria

As in classical game theory, sufficiency condition for existence of robust equilibrium can be obtained from the standard fixed-point theory which we recall next. Let $\mathcal{A}_j$ be nonempty compact convex sets and l_j^* be continuous functions such that for any fixed $(z_k)_{k \neq j}$, the function $z_j \mapsto l_j^*(z, m)$ is quasi-convex for each j. Then, there exists at least one distributionally robust pure equilibrium. This result can be easily extended to the coupled-action constraint case for robust generalized Nash equilibria.

The next section presents algorithms for computing a distributionally robust solution from the equivalent formulations above.

4.3 Learning Algorithms

Learning algorithms are crucial for finding approximate solutions to optimization and control problems. They are widely used for seeking roots of a function and for finding feasible solutions to variational inequalities. Practically, a learning algorithm generates a certain trajectory (or a set of trajectories) toward a potential approximate solution. Selecting a learning algorithm that has specific properties such as better accuracy, more stability, less-oscillatory and quick convergence is a challenging task. From the calculus of variations point of view, however, a learning algorithm generates curves. Therefore, selecting an algorithm among the others leads to an optimal control problem on the spaces of curves. Hence, it is natural to use optimal control theory to derive faster algorithms for a family of curves. Bergman-based algorithms and risk-aware version of it are introduced below to meet specific properties. We start by introducing the average regret as an evaluation metric, and then propose three learning algorithms based on Armijo's gradient flow, Bregman dynamics, and risk-aware Bregman dynamics. The convergence rate and error bound of each algorithm will be given.

4.3.1 Average Regret

One of the key approaches for error quantification of the algorithm with respect to the distributionally robust optimum is the so-called *average regret*. When the regret vanishes one gets close to a distributionally robust Nash equilibrium.

Definition 4.3. The average regret of an algorithm which generates the trajectory

$b(t) = (a(t), \lambda(t), \mu(t))$ within $[t_0, T]$, $t_0 > 0$ is

$$regret_T := \frac{1}{T - t_0} \int_{t_0}^{T} \left[\max_{b^*(t) \in \mathcal{A} \times \mathbb{R}_+ \times \mathbb{R}} \mathbb{E}_m h(b^*(t), \omega) - \mathbb{E}_m h(b(t), \omega) \right] dt$$

Here we omit the subscript j and replace $\mathcal{A}_j, a_j, h_j$ with $\mathcal{A}, a, h$ for simplicity.

4.3.2 Armijo Gradient Flow

A simple modified discrete time gradient dynamics is the so-called Armijo gradient flow [83], which is given by

$$x_{k+1} = x_k - \eta_k [\nabla^2 g]^{-1} \nabla f(x_k), \quad k = 0, 1, 2, \cdots, \tag{4.19}$$

where x_0 is the initial value, $\eta_k > 0$, g is a strictly convex function on x, ∇f is the gradient of f, and $\nabla^2 g$ is the Hessian of g with respect to x. As a corollary of Armijo's convergence theorem for the gradient method, we have the steepest descent algorithm that converges under the some hypotheses.

Corollary 4.3.1. Let f be a real-valued function defined and continuous everywhere on E^n (Euclidian $n-$space), and bounded below on E^n. For fixed x_0 define $S(x_0) = \{x : f(x) \leq f(x_0)\}$. $f \in C^1$ on $S(x_0)$ and ∇f is Lipschitz continuous on $S(x_0)$, i.e., there exists a Lipschitz constant $K > 0$ such that $|\nabla f(x_i) - \nabla f(x_j)| \leq K|x_i - x_j|$ for every pair $x_i, x_j \in S(x_0)$. If

$$x_{k+1} = x_k - \frac{1}{2K} \nabla f(x_k), \quad k = 0, 1, 2, \cdots,$$

then the sequence $\{x_k\}_0^\infty$ converges to the point x^* which minimizes f.

For distributionally robust optimization, we assume $b \mapsto \mathbb{E}_m h(b, \omega) : \mathbb{R}^{n+2} \to \mathbb{R}$ to be a concave function that has a unique global maximizer solution $b^*(t) = (a^*(t), \lambda^*(t), \mu^*(t))$, and $a^*(t)$ is a feasible action profile, i.e., $a^*(t) \in \mathcal{A}$. A continuous time analog of the Armijo gradient flow is given by

$$\begin{aligned} \frac{d}{dt} b(t) &= [\nabla^2 g]^{-1} \nabla \mathbb{E}_m h(b(t), \omega) \\ b(0) &= b_0 \in \mathbb{R}^{n+2} \end{aligned} \tag{4.20}$$

where b_0 is an initial point of the algorithm and g is strictly convex on b. The advantage is that no computation of the Hessian matrix is required as in the Newton's scheme. The drawback is that the convergence rate is low as established by the following proposition.

Proposition 4.3.2. Let $\mathbb{E}_m h(b(t), \omega)$ be a convex function for b, and b^* the solution of (4.20). For $t > 0$, we have

$$0 \leq \mathbb{E}_m h(b(t), \omega) - \mathbb{E}_m h(b^*, \omega) \leq \frac{d_g(b^*, b_0)}{t}$$

where $d_g(x, y) = g(x) - g(y) - \langle g_y(y), x - y \rangle$ is the Bregman divergence on g. Then the average regret within $[t_0, T], t_0 > 0$ is bounded above by

$$regret_T := \frac{1}{T - t_0} \int_{t_0}^{T} \mathbb{E}_m [h(b^*, \omega) - h(b(t), \omega)] dt \leq d_g(b^*, b_0) \frac{\log \frac{T}{t_0}}{T - t_0} \tag{4.21}$$

Proof. In Appendix A.3. □

The pseudocode of Armijo's gradient flow is as follows

Algorithm 1 Armijo's Gradient Flow

Input: The Armijo's gradient starting from b_0 within $[0, T]$: $(b_0, \epsilon, T, g, m, h)$

Output: b^*, $regret_T$

$b \leftarrow b_0$

while $regret_t > \epsilon$ and $t \leq T$ **do**

Compute $b(t)$ according to (4.20)

Compute $regret_t$ according to (4.21)

end while

$b^* \leftarrow b(t)$

$regret_T \leftarrow regret_t$

4.3.3 Bregman Learning

In this section we develop a fast learning algorithm for solving optimization and strategic game problems. In particular, we first introduce Maximum Principles features, and link the master Euler-Lagrange equation to a second order differential system. We then use such a system to introduce the basic Bregman algorithm and discuss its convergence rate. We provide a variation of the basic algorithm called risk-aware Bregman algorithm, which constitutes a speed-up-and-average version and reduces oscillations. We then apply the algorithm in minmax games and best-response potential games.

4.3.3.1 Maximum Principle Features

Consider the optimal control problem $\inf_{u \in U} \int_0^T \hat{l}(t, x, u)dt$ such that $u = \dot{x}$. The maximum principle is a necessary condition of optimality when the underlying function is sufficiently smooth. The adjoint variable $\dot{p} = -H_x = -\hat{l}_x$ and the

optimal control minimizes the Hamiltonian

$$H(x,p) = \inf_v \{\hat{l} + pv\}$$

that is, the Legendre transform of $-\hat{l}$ applied at the point $-p$. A closed-form expression of the optimal control can be obtained and it is generically given by $v^* = H_p(x,p)$. A necessary condition for optimality says that $H_{v^*}(v^* - v) \geq 0$ for any v, where H_v denotes the sub-differential of H. This latter variational equation can be rewritten as

$$0 \leq H_{v^*}(v^* - v) = [\hat{l}_{v^*} + p](v^* - v), \quad \forall v. \tag{4.22}$$

In particular, an interior solution v^* should solve $p = \hat{l}_{v^*}$ and the adjoint equation becomes $\dot{p} = \frac{d}{dt}(-\hat{l}_{v^*}) = -\hat{l}_x(x, v^*)$, which implies that

$$\frac{d}{dt}\hat{l}_{\dot{x}} = \hat{l}_x(x, \dot{x}) \tag{4.23}$$

The latter equation is also called Euler-Lagrange equation in the field of calculus of variations. Since the minimization is among all possible curves, this minimum principle may exhibit features that enable the investigation of faster time curves. We investigate (4.23) for a class of quantity-of-interest l. Let the family of Bregman-based Lagrangian be

$$\hat{l}(x,v) = e^{\alpha+\gamma}[d_g(x + e^{-\alpha}v, x) - e^{\beta}l(x)]$$

where $d_g(y,x) = g(y) - g(x) - \langle g_x(x), y - x\rangle$ is the Bregman divergence function for a certain smooth and convex function g.

Proposition 4.3.3. If $\dot{\gamma} = e^{\alpha}$ then the Euler-Lagrange equation (4.23) reduces to the following second order ordinary differential system

$$\ddot{x} + (e^{\alpha} - \dot{\alpha}) + e^{2\alpha+\beta} g_{xx}^{-1}(x + e^{-\alpha}\dot{x}) l_x(x) = 0 \tag{4.24}$$

which can be written as a first-order system

$$\begin{cases} \dot{x} = v, \\ \dot{v} = \ddot{x} = -(e^{\alpha} - \dot{\alpha})u - e^{2\alpha+\beta} g_{xx}^{-1}(x + e^{-\alpha}u) l_x^*(x). \end{cases}$$

Proof. In Appendix A.4. □

4.3.3.2 Basic Bregman Learning Algorithm

In view of (4.24), the Bregman algorithms for static optimization problems yields

$$\begin{aligned} \frac{d}{dt}[g_x(y(t))] &= -e^{\alpha(t)+\beta(t)} l_x(x(t)) \\ y(t) &= x(t) + e^{-\alpha(t)} v(t) \\ v(t) &= \dot{x(t)} \end{aligned} \tag{4.25}$$

where initial values are $x(0), v(0)$.

Proposition 4.3.4. Let $\mathcal{X}$ be an Hilbert space and l a convex function over $\mathcal{X}$ and $\dot{\beta}(t) \leq e^{\alpha(t)}$. Then, the Bregman learning algorithm (4.25) generates an error as

$$0 \leq l(x(t)) - l(x^*) \leq e^{-\beta(t)} c_0 \tag{4.26}$$

where $c_0 := d_g(x^*,\ x(0) + e^{-\alpha(0)}\dot{x}(0)) + e^{\beta(0)}[l(x(0)) - l(x^*)] > 0$ and x^* is the optimal value.

Proof. In Appendix A.5 □

In view of the inequality $\dot{\beta(t)} \leq e^{\alpha(t)}$, the best rate of convergence among this family of functions is obtained for $\beta(t) = \beta(0) + \int_0^t e^{\alpha(t')}dt'$, where α is a time dependence function.

Definition 4.4. *(Convergence time)*
Let $\delta > 0$ and $x(t)$ be the trajectory generated by Bregman algorithm starting from x_0 at time t_0. The convergence time to be within a ball $B(l(x^*), \delta)$ of radius $\delta > 0$ from the center $l(x^*)$ is given by

$$T_\delta = \inf_{t>t_0}(l(x(t)) - l(x^*) \leq \delta)$$

Under the assumption in Proposition 4.3.4, it takes at most $T_\delta = \beta^{-1}[\log \frac{c_0}{\delta}]$ time units for the Bregman algorithms to reach the ball $B(l(x^*), \delta)$. See Table II for detailed parametric functions on the bound T_δ.

Convergence Rate	Error Bound	Time to Reach
Triple exponential $\alpha(t) = t + e^t,\ \beta(t) = e^{e^t}$	$e^{-e^{e^t}}c_0$	$\log[\log(\log \frac{c_0}{\delta})]$
Double exponential $\alpha(t) = t,\ \beta(t) = e^t$	$e^{-e^t}c_0$	$\log(\log \frac{c_0}{\delta})$
Exponential $\alpha(t) = 0,\ \beta(t) = t$	$e^{-t}c_0$	$\log \frac{c_0}{\delta}$
Polynomial order k $\alpha(t) = \log k - \log t,\ \beta(t) = k \log t$	$\frac{c_0}{t^k}$	$\frac{c_0^{1/k}}{\delta^{1/k}}$

Table 4.1: Bregman convergence rate under different parameter settings.

Note that Proposition 4.3.4 does not require the strong convexity property often used in the proof of convergence gradient dynamics and Newton-based gradient methods. This is because the Bregman divergence is carefully designed to compensate that part as a regularizer or a penalty function.

The Bregman algorithm for distributionally robust games is given in the following proposition.

Proposition 4.3.5. Let $b \mapsto \mathbb{E}_m h(b, \omega) : R^{n+2} \to \mathbb{R}$ be a concave function that has a unique global maximizer b^*. Assume that a^* be a feasible action profile, i.e., $a^* \in \mathcal{A}$. Let α, β be two functions such that $\dot{\beta(t)} \leq e^{\alpha(t)}$. The Bregman learning algorithm is

$$\begin{aligned} &\frac{d}{dt}[g_b(b(t) + e^{-\alpha(t)}\dot{b}(t))] = e^{\alpha(t)+\beta(t)}\nabla_b \mathbb{E}_m h(b(t), \omega), \\ &b(0) \in \mathbb{R}^{n+2}, \dot{b}(0) \in \mathbb{R}^{n+2} \end{aligned} \tag{4.27}$$

where $b(0)$ is the initial point of the algorithm and g is a strictly convex function on b. Let $b(t)$ be the solution to (4.27). Then the average regret within $[t_0, T]$, $t_0 > 0$ is bounded above by

$$regret_T \leq \frac{c_0}{T - t_0} \int_{t_0}^{T} e^{-\beta(t')} dt' \tag{4.28}$$

where $c_0 := d_g(b^*,\ b(0) + e^{-\alpha(0)}\dot{b}(0)) + e^{\beta(0)}\mathbb{E}_m[h(b^*, \omega) - h(b(0), \omega)] > 0$, and $b^* = (a^*, \lambda^*, \mu^*)$ is the optimal value.

Proof. In Appendix A.6 □

In particular, for $\beta(t) = -t + e^t$ the error bound to the minimum value is

$$\frac{c_0}{t} \int_0^t e^{-\beta(t')} dt' = \frac{c_0}{t} \int_0^t e^{t'} e^{-e(t')} dt' = \frac{c_0\left(\frac{1}{e} - e^{-e^t}\right)}{t},$$

and for $\beta(t) = t$, the error bound is

$$\frac{c_0}{t}\int_0^t e^{-\beta(t')}dt' = \frac{c_0(1-e^{-t})}{t}.$$

The pseudocode of basic Bregman learning algorithm is as follows:

Algorithm 2 Basic Bregman Learning

Input: Bregman learning starting from b_0 within $[0, T]$: $(b_0, \epsilon, T, g, m, h, \alpha, \beta)$

Output: b^*, $regret_T$

$b(0) \leftarrow b_0$

while $regret_t > \epsilon$ and $t \leq T$ **do**

Compute $b(t)$ according to (4.27)

Compute $regret_t$ according to (4.21)

end while

$b^* \leftarrow b(t)$

$regret_T \leftarrow regret_t$

Figure 4.1 illustrates the advantage of algorithm (4.27) compared with the gradient flow (4.20). It plots the Bregman's regret bound $\frac{c_0}{T-t_0}\int_{t_0}^T e^{-\beta(t')}dt'$ for $\beta(t) = t$ and the Armijo's regret bound $d_g(b^*, b_0)\frac{\log\frac{T}{T}}{T-t_0}$ with an initial gap of $c_0 = 25$. The advantage of algorithms (4.20) and (4.27) is that they the Hessian of $\mathbb{E}_m h(b, \omega)$ is not required as in the case of the Newton scheme. As a corollary of Proposition 4.3.2, the regret vanishes as T grows. Thus, it is a no-regret algorithm. Algorithm (4.20) may not be sufficiently fast. Algorithm (4.27) provides a higher order convergence rate by carefully designing (α, β). The average regret decays very quickly to zero. However, it may generate an oscillatory learning trajectory with a big magnitude. The next subsection presents risk-aware algorithms that reduce the

oscillatory phase of the trajectory.

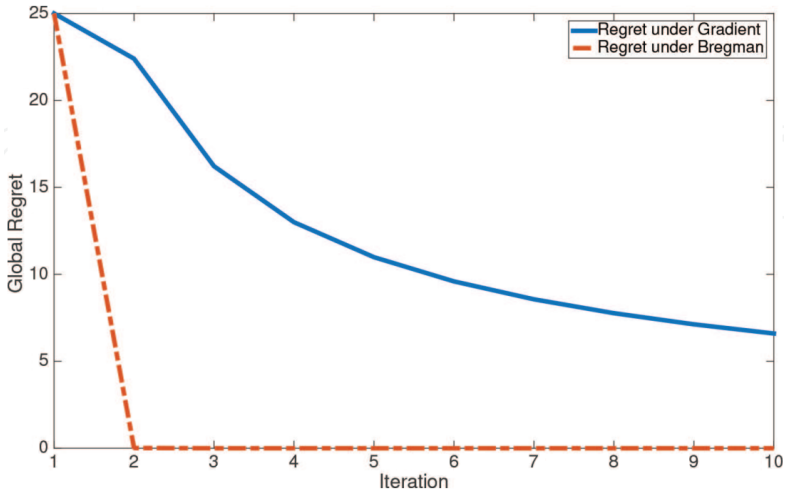

Figure 4.1: Global regret bound for Bregman vs. Armijo learning algorithms. The initial gap is set as $c_0 = 25$.

4.3.4 Risk-aware Bregman Dynamic

In order to reduce the oscillatory phase, we introduce a risk-aware Bregman dynamics which is a sort of speed-up-and-average version of (4.27) called *mean dynamics* of b. We use the average relation $\bar{b}(t) = \frac{1}{t}\int_0^t b(t')dt'$ where b solves Eq. (4.27). The time-average trajectory of the learning algorithm (4.27) generates the following mean dynamics.

$$\begin{aligned}\dddot{\bar{b}} = &-\frac{3}{t}\ddot{\bar{b}} - (e^{\alpha} - \dot{\alpha})(\ddot{\bar{b}} + \frac{2}{t}\dot{\bar{b}}) \\ &- \frac{e^{2\alpha+\beta}}{t} g_{\bar{b}\bar{b}}^{-1}(\bar{b} + (t + 2e^{-\alpha})\dot{\bar{b}} + te^{-\alpha}\ddot{\bar{b}})\mathbb{E}_m h(t\dot{\bar{b}} + \bar{b}, \omega),\end{aligned} \tag{4.29}$$

with initial values $\bar{b}(0), \dot{\bar{b}}(0), \ddot{\bar{b}}(0)$.

From the definition of $\bar{m}$ and by L'Hôpital's rule, $\bar{b}(0) = b(0)$. Moreover, $\bar{b}(t)$

and $b(t)$ share the following equations:

$$\begin{aligned} b(t) &= \bar{b}(t) + t\dot{\bar{b}}(t), \\ \dot{b}(t) &= 2\dot{\bar{b}}(t) + t\ddot{\bar{b}}(t), \\ \ddot{b}(t) &= 3\ddot{\bar{b}}(t) + t\dddot{\bar{b}}(t) \end{aligned}$$

Substituting these values in Eq.(4.27) yields the mean dynamics (4.29). The risk-aware Bregman dynamics (4.29) generates a less oscillatory trajectory due to its averaging nature. The next result provides an accuracy bound for (4.29).

Proposition 4.3.6. The risk-aware Bregman dynamics (4.29) satisfies

$$0 \leq \mathbb{E}_m[h(b^*, \omega) - h(\bar{b}(t), \omega)] \leq \frac{c_0}{t} \int_0^t e^{-\beta(t')} dt'$$

Proof. In Appendix A.7 □

Example 4.5. Let $f(x) = x \log x$ defined on $\mathbb{R}_+^*$. Then $f(1) = 0$, and the derivatives are $f'(x) = 1 + \log x$, $f''(x) = \frac{1}{x} > 0$. The Legendre-Fenchel transform of f is $f^*(\xi) = x^* = e^{\xi - 1}$. Let $(a_1, a_2) \mapsto \|a\|_2^2$ and $(a_1, a_2, \omega) \mapsto l(a_1, a_2, \omega) = 1 + \sum_{k=1}^{2} \omega_k^2 a_k^2$. The distribution of coefficient ω is unknown, but a sampled empirical measure m is considered to be similar to uniform distribution in $(0, 1]$ with 10^4 samples. We illustrate the quick convergence rate of the algorithm in a basic example and plot in Figure 4.2 the trajectories under Armijo's gradient ascent (1), basic Bregman dynamics (4.27) and riskaware Bregman dynamics (4.29). In particular, we observe that risk-aware Bregman dynamics provides very quickly a satisfactory value. In this particular setup, we observe that the accuracy of the risk-aware Bregman algorithm at $t = 0.5$ needs four times ($t = 2$) less than the basic Bregman algorithm to reach a similar level of error. It takes 40 times more

($t = 20$) than the gradient ascent to reach that level. Also, we observe that the risk-aware Bregman algorithm is less oscillatory and the amplitude decays very fast compared to the risk-neutral algorithm.

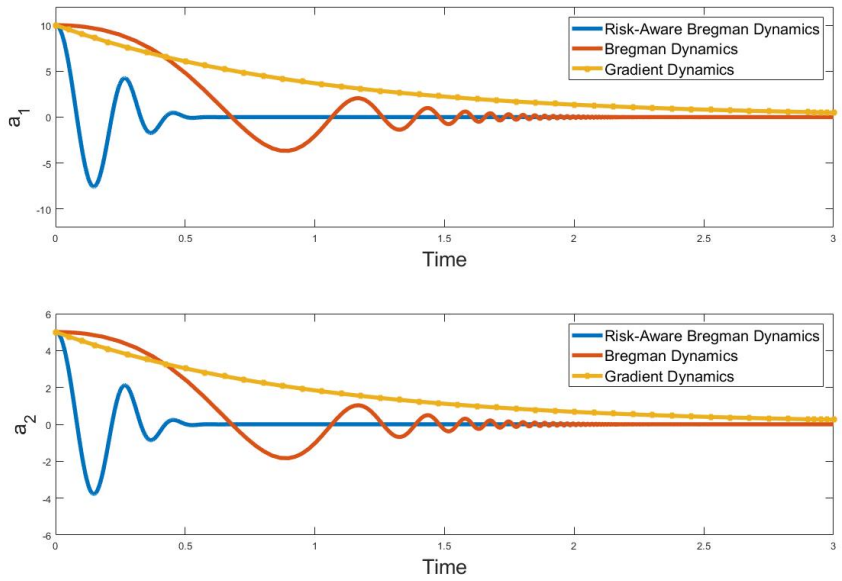

Figure 4.2: Gradient ascent vs. risk-neutral and risk-aware Bregman dynamics for $l = 1 + \sum_{k=1}^{2} \omega_k^2 a_k^2$.

4.3.5 Stochastic Bregman Learning

In this section, we introduce a stochastic Bregman learning algorithm. We first show how the Bregman dynamics apply to the distributionally robust best-response problem. In view of (4.17), the problem of player j yields

$$\inf_{x_j \in \mathcal{X}_j} \mathbb{E}_m h_j(x, \omega)$$

where the expectation is taken with respect to the random variable $\omega \sim m$. Very often the computation of the terms $\mathbb{E}_m h_j$ or its partial derivatives is challenging and depends on the structure of the distribution m. We propose swarm learning to estimate the expected gradient-like direction and then insert it to algorithm (4.27), leading to a particle swarm stochastic Bregman algorithm.

Single Particle The stochastic Bregman learning framework, which is adjusted based only on the realized integrand $h_j(b, \omega_j)$, yields

$$\frac{d}{dt}[g_{j,x_j}(x + e^{-\alpha}v)] = -e^{\alpha+\beta}h_{j,x_j}(x, \omega_j) = -e^{\alpha+\beta}[\mathbb{E}_m h_{j,x_j}(x, \omega_j) + \epsilon_j(x)]$$

where $\epsilon_j = h_{j,x_j} - \mathbb{E}_m h_{j,x_j}$ has a non-negligible variance. Now, x is a stochastic process because of the stochastic term ω and $\epsilon_j(x)$. The variance of ϵ_j is not vanishing as it is based on a single particle path discrepancy. Below, we explain how to reduce the variance by means of swarm learning.

Swarm of Particles In order to reduce variance, we introduce a swarm of virtual particles. To each player j, we associate a swarm of virtual particles ω_{jk}, $k \in 1, \cdots, N$, and assume the following Cesaro limit holds

$$\mathbb{E}h_{j,x_j}(x, \omega_j) = \lim_{N \to +\infty} \frac{1}{N} \sum_{k=1}^{N} h_{j,x_j}(x, \omega_{jk}).$$

The particle swarm-based stochastic Bregman dynamics yields

$$\begin{aligned} \frac{d}{dt} g_{j,x_j}(x + e^{-\alpha}v) &= -e^{\alpha+\beta} \frac{1}{N} \sum_{k=1}^{N} h_{j,x_j}(x, \omega_{jk}) \\ \omega_{jk} &\sim m \end{aligned} \tag{4.30}$$

Note that the error term $\epsilon_{j,N} = \frac{1}{N}\sum_{k=1}^{N} h_{j,x_j}(x,\omega_{jk}) - \mathbb{E}_m h_{j,x_j}$ has zero mean, and its standard deviation $\sqrt{\mathbb{E}[\epsilon_{j,N}^2]} = \sqrt{\frac{var[h_{j,x}]}{N^2}}$ is vanishing as N gets larger, which is much smaller than the single particle error ϵ_j.

For a realized ω, set

$$h_{j,N} = \frac{1}{N}\sum_{k=1}^{N} h_{j,x_j}(x,\omega_{jk}), \qquad x_{j,N}^* \in \arg\min_{x_j} h_{j,N}(x,\omega)$$

Let $x \mapsto h_N(x,\omega)$ be a best-response potential function for the game in strategic form $\{\mathcal{J}, (\mathcal{X}_j, h_{j,N})_{j\in\mathcal{J}}\}$, that is

$$\arg\min_{x_j} h_N(x,\omega) \subseteq \arg\min_{x_j} h_{j,N}(x,\omega), \quad \forall j \in \mathcal{J}$$

Then by applying the mean dynamic (4.29) to the distributionally robust game, the particle swarm Bregman algorithm (4.30) gives the output $x_N(t)$ that satisfies

$$0 \leq h_N(x_N(t),\omega) - h_N(x_N^*,\omega) \leq e^{-\beta(t)} c_{0,N}$$

where $c_{0,N} = d_g(x_N^*, x_0 + e^{-\alpha(t_0)}\dot{x}_0) + e^{\beta(t_0)}(h_N(x_0,\omega) - h_N(x_N^*,\omega))$, $x_0 = x(t_0)$. Thus, the average regret under risk-aware particle swarm Bregman learning is

$$regret_T \leq \frac{c_{0,N}}{T - t_0}\int_{t_0}^{T} e^{-\beta(t)} dt$$

This provides an approximated distributionally robust equilibrium when N is sufficiently large.

4.4 Illustrative Examples and Simulations

In this section, we apply the proposed learning algorithms in three scenarios. First, we show the success of Bregman algorithm in a simple minimax game case. It converges with exponential decay, while the standard gradient descent keeps oscillating and fails to find the equilibrium. Second, we examine the stochastic Bregman learning algorithm on a multi-modal optimization problem. It is of practical interest when the loss function is not differentiable or its mathematical expression is not available. Third, we test limitations of the proposed algorithms on a robust game with non-convex non-concave objective functions.

4.4.1 Minimax Game Case

Consider two agents and let $x = (x_1, x_2) \in \mathbb{R}^2$ be their actions. Agent 1 is the attacker who chooses $x_1 \in \mathbb{R}$ and agent 2 is the defender who chooses $x_2 \in \mathbb{R}$. They have opposite objectives and form a zero-sum minimax game. We set the loss function as $l(x_1, x_2) = x_1 x_2$ and the optimization problem as $\min_{x_1} \max_{x_2} l(x_1, x_2)$. Note that the objective is not convex for this minimax problem. However, it is clear that $(0, 0)$ is an equilibrium as it satisfies

$$l(0, x_2) = l(x_1, 0) = 0 \quad \forall x_1, x_2$$

Here, we set the initial values as $(0.9, 0.5)$ and see if the algorithms can find the Nash equilibrium.

Failure of standard gradient descent We observe the classical gradient flow leads to periodic solutions (as the sine function is a fundamental solution) with

cycling behavior. In particular, the amplitude of oscillation of the classical gradient does not decrease over time. It leads to a limit cycle on an ellipsoid with a non-decreasing amplitude. Thus, the classical gradient flow fails to find the Nash solution $(0, 0)$ as illustrated in Figure 4.3. For any fixed x_2, the function l is quasi-convex in x_1 but not strictly convex. The Hessian is degenerated for any fixed x_2. The classical gradient descent/ascent algorithm fails to find the saddle point and leads to a cycling behavior on an ellipsoid. Indeed,

$$\frac{d}{dt}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} x_2 \\ -x_1 \end{pmatrix}$$

This equation is equivalent to the second order dynamics $\frac{d^2}{dt^2}x_1 + x_1 = 0$, which leads a combination of trigonometric functions cosine and sine. Thus, there is an oscillatory regime and no convergence to the saddle point $(0, 0)$. The amplitude of oscillation is in the order of the initial magnitude. We will see below how to construct an algorithm that will reduce the amplitude and approximates better the equilibrium.

Success of Bregman Algorithm We use the risk-aware Bregman algorithm to find the saddle point of the function $(x_1, x_2) \mapsto l(x_1, x_2) = x_1 x_2$ with a penalty function g. We set $g_{xx} = I$, $\alpha(t) = t$, and $\beta(t) = e^t$. In Figure 4.4 we plot the evolution of strategies and payoff values. We observe rapid convergence with a smaller amplitude than the classical gradient algorithm.

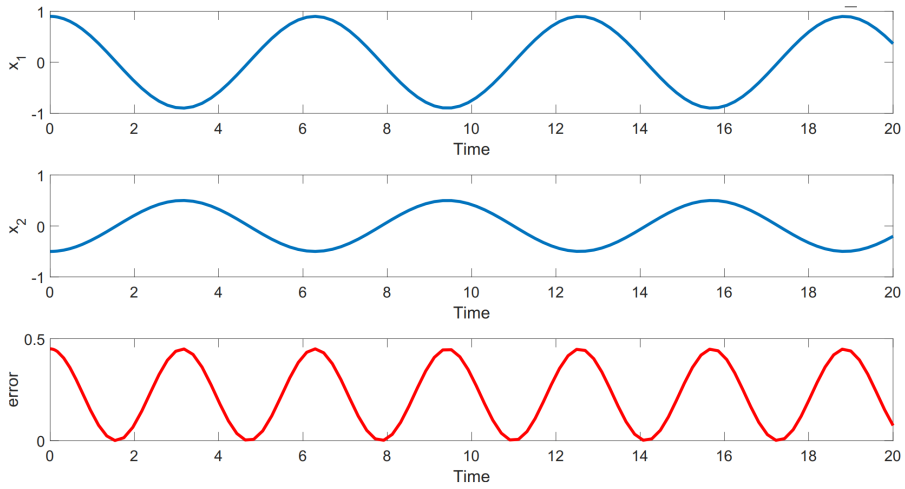

Figure 4.3: The amplitude of oscillation of the standard gradient does not decrease. The classical gradient flow fails to find the equilibrium point $(0, 0)$.

4.4.2 Non-convex Multi-modal Robust Optimization

We set the objective function

$$l_i(a, \omega) = \log\left(20 + \omega^2 - \sin(a_1)\sin(a_2)\sqrt{a_1 a_2}\right)$$

over the action space $[0, 10]^2$. It has multiple local extrema (Figures 4.5). The function is non-convex and non-concave, and thus does not fulfill the conditions in Proposition 4.3.4. The multi-modal loss function has a robust equilibrium at $(7.9, 7.9)$. We observed that the basic Bregman algorithm (4.25) stuck in a local optimum $(5.1, 7.9)$, while the particle swarm Bregman algorithm (4.30) behaves well and converges at the robust equilibrium (Figure 4.5 right). This opens the application of stochastic Bregman learning in non-convex settings.

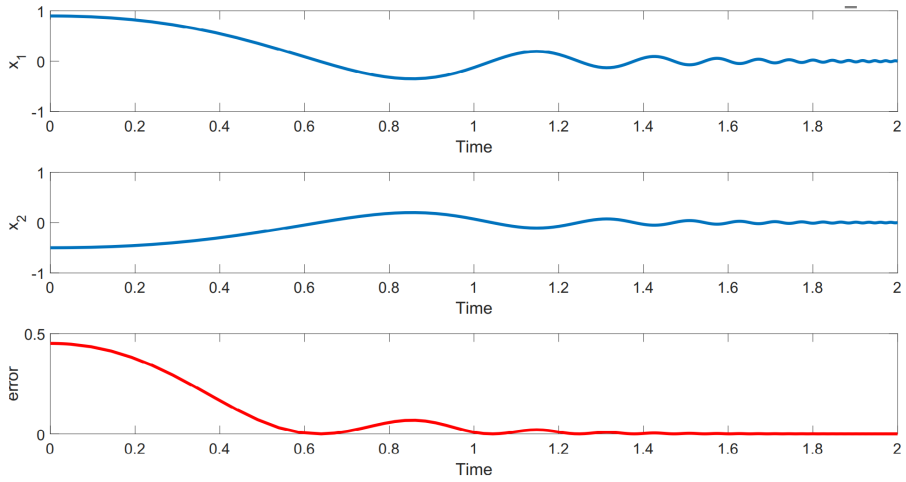

Figure 4.4: The proposed scheme leads rapidly to a smaller amplitude of oscillation. It converges within two time units.

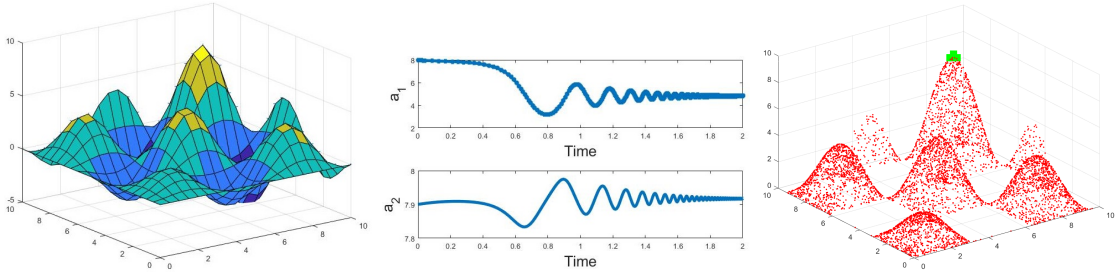

Figure 4.5: Left: The non-convex non-concave objective function l_i with multiple local optima. Middle: The basic Bregman algorithm stuck in a local optimum $(5.1, 7.9)$. Right: The particle swarm Bregman algorithm converges to the global optimum $(7.9, 7.9)$.

4.5 Experiments

In this section, we implement specialized Bregman algorithms for neural network models and compared the performance with other widely known algorithms. The problem is much difficult than classical optimization tasks. The challenges lie in non-convexity of the loss function and high dimensionality of the model parameters. In most cases, the number of local optima is huge or even infinite. For this reason, we typically search for suboptimal solutions instead of a global optimum. This part

is based on our work "Bregman Learning for Generative Adversarial Networks" [85].

Stochastic gradient descent (SGD) [86, 87] is the most popular optimization algorithm in deep learning. The objective function $L(x)$ is minimized by gradually going downhill towards the opposite direction of the gradient of L with respect to its parameters x, and the step size is controlled by a learning rate η:

$$x_{t+1} = x_t - \eta \cdot \nabla_{x_t} L(x_t)$$

The downhill direction is estimated by taking the average gradient on a minibatch of M samples randomly drawn from the training set. For classification problems, the gradient is estimated by

$$\mathbb{E}\nabla_x L(x) \approx \frac{1}{M} \sum_{i=1}^{M} \nabla_x L(x; \mathrm{x}^{(i)}; \mathrm{y}^{(i)})$$

where $\mathrm{x}^{(i)}$ is the training data with corresponding label $\mathrm{y}^{(i)}$. However, the convergence rate of gradient descent cannot exceed $O(\frac{1}{t})$ after t iterations [88], and it may become even worse in practice. SGD may be trapped in local optima, and has trouble in the case of high curvature and noisy gradients. The trajectory tends to be oscillating in the ravine region where the gradient is much more higher in one dimension than in others. Momentum [89] is a method to reduce trajectory oscillations and accelerate SGD. The update rule is:

$$\begin{aligned} x_{t+1} &= x_t + v_t \\ v_{t+1} &= \gamma v_t - \eta \cdot \nabla_{x_t} L(x_t) \end{aligned} \tag{4.31}$$

The velocity term $v = \dot{x}$ stores an exponential decay average of the past gradients, which introduces an inertia force to push a particle in the parameter space. The updates for dimensions with consistent downhill are accelerated, while the updates are slowed down if the gradient directions change frequently. This reduces oscillation and improves convergence rate. The Bregman learning algorithm is a configurable optimization approach. We will show that SGD [87] and Momentum [89] are two special cases in our scheme.

Derive from the second order ordinary differential equation (ODE) defined in (4.24) and apply the gradient $\nabla_x L$, we have

$$\frac{d}{dt}\left(\frac{\partial g}{\partial x}(x + e^{-a(t)}\dot{x})\right) = -e^{a(t)+b(t)}\nabla_x L$$

By taking $g(x) = \frac{1}{2}c_1 x^T x$, $v = \frac{dx}{dt}$, it can be split into two first order ODEs,

$$\begin{aligned} \dot{x} &= v \\ \dot{v} &= -(e^a - \dot{a})v - e^{2a+b}c_1^{-1} \cdot \nabla_x L \end{aligned} \tag{4.32}$$

Under the configuration that $c_1 = 1 - \gamma$, $a(t) = \ln(1-\gamma)$, $b(t) = \ln \eta - a(t)$,

$$\begin{aligned} \dot{x} &= v \\ \dot{v} &= -(1-\gamma)v - \eta \cdot \nabla_x L \end{aligned} \tag{4.33}$$

Discretize it over finite time interval Δt, we get

$$\begin{aligned} x_{t+1} &= x_t + v_t \Delta t \\ v_{t+1} &= v_t + (-(1-\gamma)v_t - \eta \cdot \nabla_x L)\Delta t \end{aligned} \tag{4.34}$$

which is equivalent to Momentum in (4.31) when $\Delta t = 1$.

If we set $\gamma = 0$, it reduces to SGD.

$$\begin{aligned} x_{t+1} &= x_t + v_t \\ v_t &= -\eta \cdot \nabla_x L \end{aligned} \tag{4.35}$$

By setting $a(t)$, $b(t)$ and $g(x)$, we get different specialized versions of Bregman learning algorithm. Some of the configurations are listed in Table 4.5.

$a(t)$	$b(t)$	Convergence Rate
t	e^t	double exponential
0	t	exponential
$\ln k - \ln t$	$k \ln t$	polynomial (order k)
0	$\ln \eta$	*same as Gradient Descent [86]
$\ln(1-\gamma)$	$\ln \eta - a(t)$	**same as Momentum [89]

Table 4.2: Bregman learning algorithm under different configurations. $g(x) = \frac{1}{2}c_1 x^T x$, $^*c_1 = 1$, $^{**}c_1 = 1 - \gamma$.

4.5.1 Supervised Learning (Image Classification)

For the first application, we apply Bregman learning to train a neural network classifier, and compare it with six most popular algorithms in deep learning. The generalization performance is evaluated by calculating the test error.

The MNIST dataset [90] contains 70000 labeled images of handwriting digits (Figure 4.6). Our goal is to train a classifier to predict the digit for each handwriting. We split it into three sets: 55000 for training, 5000 for validation and 10000 for testing. The hyperparameters are selected by cross validation.

The classifier is a standard softmax regression model with three layers. The neuron's input is a vector of 784 dimensions flattened by a 28×28 image. Then it

Figure 4.6: An example of handwriting digits

goes through a linear layer with weights W and biases b. Finally it is activated by a softmax function and output a probability distribution over 10 digits. The loss function is cross entropy between training label and the predicted probabilities.

We use the same model but different algorithms to optimize the model parameters W and b. The algorithms under comparison are: Stochastic Gradient Descent (SGD) in [86], Momentum in [89], Adaptive Moment Estimation (Adam) in [91], RMSProp in [92], Adadelta in [93], Adagrad in [94], and our Bregman-based optimization algorithm. The configuration in this experiment is $t_0 = 3$, $\Delta t = 1$, $a(t) = \ln k - \ln t$, $b(t) = k \ln t$ with $k = 2$. For other algorithms, the fine-tuned hyperparameters are listed in TABLE 4.5.1.

SGD	$\eta = 0.5$
Momentum	$\eta = 0.5, momentum = 0.9$
Adam	$\eta = 0.3, \beta_1 = 0.95, \beta_2 = 0.999, \epsilon = e^{-8}$
RMSProp	$\eta = 0.02, momentum = 0.9, \alpha = 0.9, \epsilon = e^{-10}$
Adadelta	$\eta = 0.5, \rho = 0.95, \epsilon = e^{-8}$
Adagrad	$\eta = 0.5, initial\_accumulator\_value = 0.1$

Table 4.3: Hyperparameters

The algorithms are evaluated in 100 iterations by checking the classifier's accuracy on test data. Figure 4.7 shows that Bregman learning algorithm achieves

the highest performance. For stochastic optimization, the sample size varies from 1000 to 55000. Numerical results of the average classification error in 100 iterations are listed in Table 4.5.1. Compared to existing optimization algorithms used

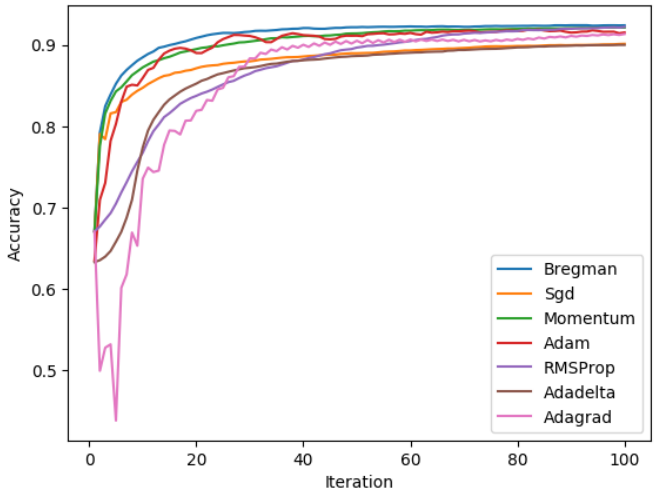

Figure 4.7: Comparison of seven optimization algorithms

in deep learning, the Bregman-based approach quickly converges and achieves state-of-the-art generalization performance.

sample size	1000	2000	5000	10000	55000
Bregman	0.1128	**0.1002**	**0.0926**	**0.0909**	**0.0897**
SGD	0.1279	0.1236	0.1216	0.1211	0.1198
Momentum	**0.1060**	0.1015	0.0976	0.0984	0.0975
Adam	0.1152	0.1211	0.1093	0.1065	0.1021
RMSProp	0.1384	0.1366	0.1310	0.1294	0.1286
Adadelta	0.1442	0.1425	0.1393	0.1399	0.1390
Adagrad	0.1552	0.1463	0.1456	0.1454	0.1440

Table 4.4: Average Classification Error

4.5.2 Unsupervised Learning (Image Generation)

The second example is an unsupervised learning task, where we are going to learn the distribution of real data and generate artificial samples which can pass the Turing test.

Convolutional neural network (CNN) with multiple hidden layers is a powerful tool to learn a hierarchy of representations from image data. The huge amount of non-linear hidden units provide sufficient model capacity, but also result in general non-convexity. The cost function surfaces contain plenty of non-convex structures such as cliffs, ravines, plateaus, local minima and high cost saddle points. In practise, the learning rate for gradient descent should be well tuned to make small and careful moves. Methods based on higher-order gradients could cause instability around saddle points. Although the convergence analysis assumes convex loss function, we found that our method still outperforms others in the non-convex cases after tuning the parameters. We show the effectiveness of Bregman algorithm for training deep neural networks.

For fair comparison, we adopt consistent model structure as in the previous literature [95], in which SGD [86] and Adam [91] were used for training. The deep generative model has two CNNs, serving as the generator G and the discriminator D. Parameterized objective function has been given in Example 1. The CNNs stack four hidden convolution/deconvolution layers, each followed by batch normalization [88] (batch size = 64), ReLU/LReLU activation [96, 97], and finally produce sigmoid outputs.

The two CNNs are optimized in an adversarial way: D tries to distinguish the synthesis data from the real one, while G tries to mimic the real data distribution and produce fake samples to fool the discriminator. The training is like a pingpong

game, and gradually converge to a saddle point. At each iteration, a random noise z of 100 dimensions is feed into the generator to produce synthesis samples $G(z)$. We found uniform distribution $U(-1, 1)$ gains better results than Gaussian noise.

The model was trained by SGD, Adam and Bregman algorithm respectively, with hyperparameters in Table 4.5.2. For Bregman algorithm, we use the same configuration as in the last experiment but slightly change the setting of t. We found too large t could lead to oscillation, while set a maximum bound on t helped stabilize training. To deal with the rugged cost function surface with general non-convexity, we use a small time interval Δt for discretization.

*SGD	$\eta = 0.1$
*Adam	$\eta = 0.0002, \beta_1 = 0.5, \beta_2 = 0.999, \epsilon = 10^{-8}$
*Bregman	$t_0 = 0.1, t_{max} = 2.0, \Delta t = 10^{-3}$
**SGD	$\eta = 0.005$
**Adam	$\eta = 0.0002, \beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}$
**Bregman	$t_0 = 0.5, t_{max} = 3.0, \Delta t = 10^{-4}$

Table 4.5: Hyperparameters (*MNIST, **CelebA)

We test our approach on two datasets: MNIST handwriting (70,000 images) [90] and celebA human face (200K photos) [98]. In the first 100 iteration (first row of Figure 4.8), the generator outputs fragmented and blurring handwriting. Then digits emerges (middle), and eventually get matured and perfectly indistinguishable from the real one (compare the bottom right image with Figure 4.6).

In the beginning of our second experiment (first row of Figure 4.9), only the most salient features are generated. SGD (left) draws only eyes, Adam (middle) has eyes and mouth, while Bregman (right) also includes the contour of human face. During the evolution, SGD collapses to a single point and keeps producing identical samples (left column of Figure 4.9). This is due to the weakness of SGD

to escape from local optima in the parameter space.

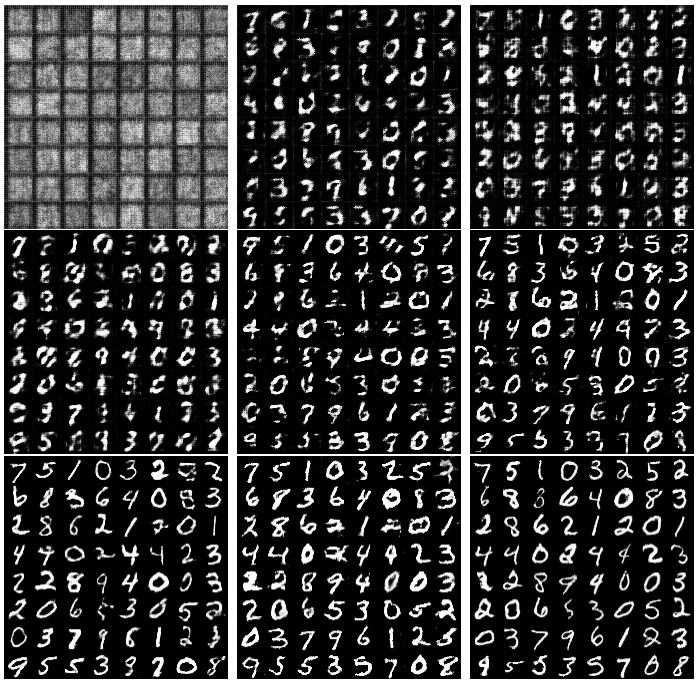

Figure 4.8: Comparison of generated samples on MNIST. From top to bottom, iteration: 100, 1100, 27200. From left to right, algorithm: SGD, Adam, Bregman

Figure 4.9: Comparison of generated samples on CelebA. From top to bottom, iteration: 100, 2000, 31500. From left to right, algorithm: SGD, Adam, Bregman

4.6 Discussion

We have introduced distributionally robust games with continuous action space and a possible adversarial modification of the uncertainty. The problem is formulated with a notion of divergence between two measures: the modified worst-case measure and the true measure associated to the uncertain environment. We proposed several learning algorithms based on Bregman discrepancy to speedup the computation of robust equilibria.

The Bregman learning algorithm is a configurable general approach. When the payoff is convex or concave, it achieves double exponential convergence rate. By choosing different parametric settings, it can be successfully implemented in various tasks with non-convex non-concave objectives. In the experiments, both qualitative and quantitative evaluation results show the advantage of our algorithm in training neural network models.

In this chapter, the objectives are implemented with simple functions. In order to extend the framework to solve more complex problems, one direction is to design appropriate neural network models to learn the players' objectives. The simple function approximation lemma tells us that for any Lebesgue measurable function h_j, we can find step functions which are arbitrarily close together. The set of simple functions is dense in the set of measurable functions. It leads to interactive neural networks as the objective functions are interdependent through the actions of the players. The following chapters will explore this idea and its relationship with deep generative models.

Chapter 5

Learning Generative Models

Deep generative models are powerful but difficult to train due to their instability, model collapse, and saturation problems. This chapter presents an interplay between generative modeling and game theory. We introduce a distributionally robust game (DRG) framework to train generative models, and use Wasserstein metric to measure the discrepancy between generated and real data distributions. Two concepts are brought together within this framework: robust game and generative modeling. In this game, two types of agents work in opposite directions and form a robust optimization problem. The attacker computes adversarial perturbations of *real samples* generate *hard samples*. They are supposed to have maximum distance from the model distribution, and can be easily distinguished from the generator's outputs. On the other side, the defender updates the generative model to fit for those *hard samples* and generates *fake samples*. They are supposed to distributed similar to the real data so that the discrepancy between model and data distributions is minimized. Those agents are trained alternatively based on the outcome of others. The best strategy is learnt by dynamically optimizing on the worst-case scenario

created by the other side. Convergence is reached when no agent can improve its performance. We develop learning algorithms to find the distributionally robust Nash equilibrium.

Compared with Kullback-Leibler divergence that are widely used in many GAN models, Wasserstein distance has a weaker topology and provides a usable gradient almost everywhere. It offers continuous loss function under mild assumptions, and therefore is able to strengthen the training stability. However, computing the Wasserstein distance involves solving an optimal transport problem, which is very challenging for high dimensional data. We propose an equivalent problem that reduces considerably the curse of dimensionality of the problem, and provide algorithms to approximate the Wasserstein distance. Experiments are carried out on real datasets to test both shallow and deep generative models. Both theoretical merits and practical advantages of our approach are examined. Performance evaluation shows the training process is stable and converges fast. Our model can produce visual pleasing *fake samples* which are highly resemble to the real data.

This chapter is based on a long-standing work in the field and it refers to our papers "Distributionally Robust Games: f-Divergence and Learning" [99] and "Distributionally Robust Games: Wasserstein Metric" [100]. The main contributions are summarized as follows:

- We propose a novel game theory framework to learn generative models. To the best of our knowledge, this is the first work connecting distributionally robust game with generative modeling.

- We develop learning algorithms to solve the distributionally robust equilibrium under Wasserstein metric. The complexity of the problem is reduced by

introducing an equivalent optimization problem, in which the Wasserstein distance is computed on low-dimensional space.

- We provide practical implementations of our framework to train deep generative models. Motivated by the recent progress in image synthesis, we apply the proposed model to the challenging task of unsupervised image generation. The evaluation results confirm the advantage of our approach in non-trivial applications.

5.1 Introduction

Artificial intelligence has achieved great progress in the field of supervised learning, including the tasks of classification, recognition and prediction. Apart from these, people want to learn the full distribution of data and help computers to better understand the real world. Generative modeling is an approach to learn such representations. Given a number of observations, it learns the underlying distribution of data in an unsupervised manner and produce new data in the test phase. Besides generating new samples from scratch, it can also take in conditional information to generate data with user-specified properties. In generative modeling, the generated samples are supposed to be similar or indistinguishable from the real ones and have enough diversity. Generative models are powerful tools for many tasks such as signal denoising, image inpainting, data synthesis and semi-supervised learning.

Classical generative modeling is often based on maximum likelihood estimation, e.g., Restricted Boltzmann Machines. Most modern models are run on deep neural networks, which have similar architectures with those used in supervised learning.

They extract distinctive features of data and use these fundamental understandings to build the model. Training deep generative models is highly unstable and slow to converge. The high dimension of training data and complex structure in objective functions lead to many problems, for example, exploding or diminishing gradient, mode collapse, and poor sample quality. Moreover, the model should have strong generalization power to produce diverse new samples instead of just memorizing the training set.

Variational Auto-Encoders (VAEs) [50] and Generative Adversarial Networks (GANs) [53] are the most popular deep generative models today. VAE involves an inference network to explicitly formulate the posterior distributions of laten variables, and maximizes a lower bound on the likelihood. It has a nice theory, but practically, VAE samples suffer from blurry due to the noise term in their model density functions [101]. GANs alternatively train a generative network and a discriminative network with opposite objectives. The loss functions are defined by information-based metric like Kullback-Leibler and Jensen-Shannon divergence. Despite its great success in producing visual pleasing samples, the training process is unstable. It needs to carefully keep the balance between updating those two networks to avoid gradient saturation [102]. To deal with this problem, Martin Arjovsky switches to the Earth Mover (EM) distance and proposed WGAN [103]. Their model involves another neural network to estimate EM, whose weights are clipped to enforce the Lipschitz constraint. Variations under this framework include conditional GAN [104] that generate samples conditioned on class labels, LAPGAN [105] that generates images in a coarse-to-fine fashion, WGAN-GP [57] that uses gradient penalty to stabilize the training process.

Different from VAE and GAN models, Generative Moment Matching Networks

(GMMN) [106] do not need a second network. The generative model is trained by minimizing the maximum mean discrepancy (MMD), where the objective is evaluated by matching all moments of the statistics between real and fake sample distributions. By using kernel tricks, explicit computation of those moments are not required. Recently, Aude Genevay [107] proposed a method to learn with Sinkhorn divergence, which is a mixture of MMD and the optimal transport loss [96]. The references mentioned above do not examine Wasserstein-based distributionally robust games.

We introduce a new game-theoretic framework for generative modeling. We formulate the problem as a distributionally robust game (DRG) under uncertainty and offer a corresponding distributionally robust equilibrium concept. In this game there are two groups of players with opposite objectives. Each player works on a continuous action space to optimize the distributionally worst-case payoff. Our model differs from the distribution-free robust game framework proposed by [20, 81]. In their approach, the uncertainty set needs to be pre-specified by the decision makers in advance, while in our approach any alternative distribution within a certain.

Another issue is how to define the distance between two distributions, i.e., the similarity between real and fake sample sets. Instead of information based metric such as KullbackLeibler (KL) divergence, we use Wasserstein metric to measure the distance between two distributions. It is a real distance and has finer topology in the parameter space, which provides better gradients and therefore improves the stability of the optimization algorithm. However, computing Wasserstein distance involves solving an optimal transportation problem, which is nontrivial. Marco Cuturi [108] adds an entropy regularization term to the original problem and

switch to calculating the Sinkhorn distance. Martin Arjovsky [103] works on the Kantorovich-Rubinstein dual problem, and trains a neural network to estimate the cost. In our framework, this task is given to the defenders, who explore the Wasserstein ball of real data to generate adversarial samples for attackers. Using Moreau-Yosida regularization [109, 110], we transform the optimization problem based on Wasserstein distance into an optimization under Euclidean distance, and solve it on much lower dimensions.

To train deep generative models for image synthesis, we implement the attackers and defenders with convolutional neural networks. Since the data has very high dimension, we add an encoder network to learn meaningful feature representations and embed them into the model. The Wasserstein distance is computed on the latent space.

5.2 Problem Formulation

5.2.1 GAN as a Minimax Game

In unsupervised learning, people want to learn from the distribution of training data without labeled information. Denote m as the *real data* distribution and $\tilde{m}$ as the *fake data* distribution over space Ω. Learning generative models aims at minimizing the statistical distance between these two distributions. The GAN problem involves minimizing $\inf_{\tilde{m}} D_f(\tilde{m} \parallel m)$ from the generator perspective who acts on $\tilde{m}$. However, m is not a given profile, so it needs to be estimated from the training samples. This leads to a two-layer learning problem: the discriminator tries to maximize the gap so as to distinguish real and fake samples, while the generator learns to minimize it so that the fake samples are similar to the real ones.

For a convex and proper function f the Legendre-Fenchel duality holds $f^{**} = f$, where $f^*(\xi) = \sup_x[\langle x, \xi\rangle - f(x)]$. We use Legendre-Fenchel duality to formulate the GAN's optimization problem

$$\begin{aligned}
\inf_{\tilde{m}} D_f(\tilde{m} \parallel m) &= \inf_{\tilde{m}} \int_\Omega f^{**}\left(\frac{d\tilde{m}}{dm}\right) dm - f(1) \\
&= \inf_{\tilde{m}} \int_\Omega \left(\sup_\xi \left\langle \xi, \frac{d\tilde{m}}{dm}\right\rangle - f^*(\xi)\right) dm - f(1) \\
&= \inf_{\tilde{m}} \left(\int_\Omega \sup_\xi \langle \xi, d\tilde{m}\rangle - \int_\Omega f^*(\xi) dm\right) - f(1) \\
&\geq \inf_{\tilde{m}} \sup_\xi \left(\mathbb{E}_{\tilde{m}} \xi(\omega) - \mathbb{E}_m f^*(\xi(\omega))\right) - f(1)
\end{aligned} \tag{5.1}$$

The optimal structure for ξ is $\xi^\star(\omega) = f'(\frac{d\tilde{m}}{dm}(\omega))$. We assume that the optimal $\xi^\star$ can be parameterized by and represented as $\xi^\star(\omega) = h_f(V(a_1, \omega))$ for some functions h_f and V, where $V(a_1, \omega) : \mathcal{A}_1 \times \Omega \to \mathbb{R}$, and $h_f : \mathbb{R} \to Dom(f^*), Dom(f^*) = \{x \mid f^*(x) \in \mathbb{R}\}$. The measure m is re-parameterized with m_{a_2}, $a_2 \in \mathcal{A}_2$. Thus,

$$\inf_{\tilde{m}} D_f(\tilde{m} \parallel m) = \inf_{a_1} \sup_{a_2} \mathbb{E}_{\tilde{m}_{a_1}} h_f(V) - \mathbb{E}_{m_{a_2}} f^*(h_f(V)) - f(1) \tag{5.2}$$

The optimization problem $\inf_{\tilde{m}} f(\tilde{m} \parallel m)$ is formulated as a zero-sum game with two adversarial players. One acts as a discriminator to find the maximum gap between m and $\tilde{m}$, while the other acts as a generator that tries to synthesize indistinguishable samples to fool the discriminator. Since the payoff function is not deterministic and dependent on the other side, each player optimizes its worst-case performance. The GAN problem is therefore equivalent to a minimax game, and

the corresponding optimization problem is as follow

$$(P)\begin{cases}\inf\limits_{a_1\in\mathcal{A}_1}\sup\limits_{a_2\in\mathcal{A}_2} L(a_1,a_2),\\ L(a_1,a_2)=\mathbb{E}_{\tilde{m}_{a_1}}h_f(V(a_1,\omega))+\mathbb{E}_{m_{a_2}}(-f^*)(h_f(V(a_1,\omega)))-f(1)\end{cases}\tag{5.3}$$

where $\mathcal{A}_1$ denotes the decision space of the generator and $\mathcal{A}_2$ is the decision space of the discriminator.

Example 5.1. Let $f(x)=x\log x-(1+x)\log(1+x)$. Then the derivative of the function f is $f'(x)=\log(1-\frac{1}{1+x})$ and $f''(x)=\frac{1}{x(1+x)}>0$. This means f is a convex function, and $f(1)=-2\log 2$. We then apply the Legendre transform

$$f^*(\xi)=\sup_x[\langle x,\xi\rangle-f(x)]=-\log(1-e^{\xi})$$

The parameterized objective function is now

$$L(a_1,a_2)=\mathbb{E}_{\tilde{m}_{a_1}}h_f(V)+\mathbb{E}_{m_{a_2}}\log(1-e^{h_f(V)})-f(1)$$

If $h_f=\log$, then

$$L(a_1,a_2)=\mathbb{E}_{\tilde{m}_{a_1}}\log(V)+\mathbb{E}_{m_{a_2}}\log(1-V)-f(1).$$

In order to provide training data with enough diversity, a particular sampling process was considered in [53]. Real data is randomly sampled from the training set $x_{real}\sim m$; fake data is first sampled from a noise distribution z then fed into the generative model $x_{fake}=G(z)\sim\tilde{m}$. The discriminator takes in both real and fake samples, and outputs two probabilities $V=D(x_{real})$ and $D(x_{fake})$ for a_1,a_2

respectively. In this game, two opponent agents optimize on different objective functions. It is a zero-sum game with no cooperation. The attacker $G(z)$ tries to fool the defender D by minimizing $\log(1 - D(G(z)))$, which means incorrectly predicting the fake sample as real. The defender optimizes the discriminator by maximizing the sum of two losses $\log(D(x_{real})) + \log(1 - D(G(z)))$, which means correctly predicting both the real and fake samples. There is a sort of tradeoff: maximizing the first term encourages to correctly identify the real samples, while maximizing the second term encourages correctly finds out the fake ones. The GAN model is trained by solving Nash equilibrium in a minimax game. The formulation is robust in the sense that the attacker tries to sneak into the real data distribution while facing the supervision of the defender.

5.2.2 The DRG framework for Generative Modeling

Building on the above observation, we develop a distributionally robust game framework to learn generative models. Remember that the standard best response problem of player j: $\inf_{a_j \in \mathcal{A}_j} l_j(a_j, a_{-j})$ becomes a minimax robust response problem $\inf_{a_j \in \mathcal{A}_j} \sup_{\omega \in \Omega} l_j(a_j, a_{-j}, \omega)$ when the objective function l_j depends on an uncertain state ω. In DRG, one acts with an uncertainty set that consists of distributions chosen by other players. Specifically, the uncertain state ω is a random variable with distribution m, which is not known. The only available is some incomplete observation of m. We assume there is a set of distributions lie in the neighborhood of m: $B_\rho(m)$, then the distributionally robust best response is

$$\inf_{a_j \in A_j} \sup_{m' \in B_\rho(m)} \mathbb{E}_{\omega' \sim m'} l_j(a_j, a_{-j}, \omega') \tag{5.4}$$

We choose the uncertain set as probability distributions within a Wasserstein ball of radius ρ from m (as shown in Figure 5.1)

$$B_\rho(m) = \{m' \mid W(m, m') \leq \rho\} \tag{5.5}$$

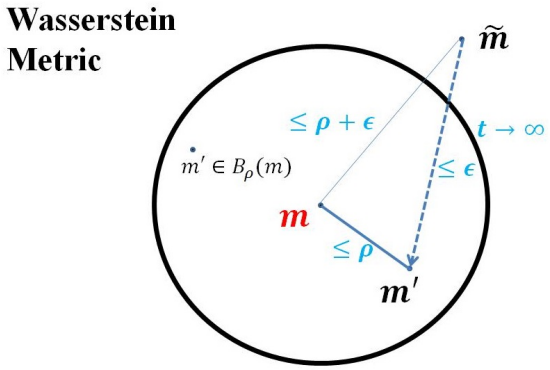

Figure 5.1: Uncertainty set defined by a Wasserstein ball

Recall that in distributionally robust games the players have conflicting objectives and are optimizing their worst-case performance. The existence of robust Nash equilibrium has been proved in Chapter 4. Reaching the robust equilibrium means all players achieve the minimum loss in their worst-case scenario. Next we formulate the generative modeling problem into the DRG framework. As depicted in Figure 5.2, there are two types of players in this game. The attackers train the generative model $G_{\theta_a}(z)$ to produce *fake samples* $\tilde{x}_i \sim \tilde{m}$, which are supposed to be similar to or indistinguishable from the real ones. z is a low dimension random variable feed into the generator, and θ_a denotes the model parameter. The defenders explore the Wasserstein neighborhood of m and compute perturbations of real data to produce *hard samples* $x'_i \sim m'$. The *hard samples* are supposed to have maximum

distance from the model distribution and can be easily distinguished. The objective function is defined by the discrepancy $D(\tilde{m} \parallel m')$, where $\tilde{m}$ is the distribution of *fake samples* produced by the attacker, and m' is the distribution of *hard samples* produced by the defender. Since the real distribution m is unknown and we only have an observation dataset $\{x_1, \ldots, x_N\} \subseteq \mathbb{R}^d$, the optimization is performed by iteratively updating the generative model as well as the hard sample distribution m'. It represents an approximation of m within some bounded uncertain set $B_\rho(m)$.

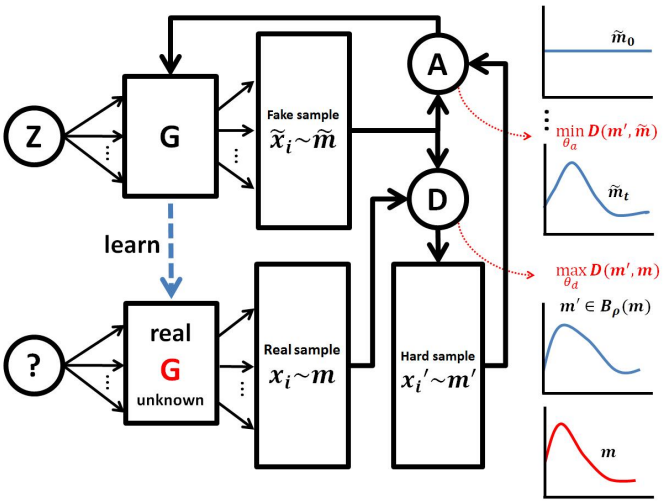

Figure 5.2: Distributionally robust game framework for generative modeling

As displayed in the right column of Figure 5.2, the fake samples are initially drawn from an arbitrary distribution, e.g., $\tilde{m}_0$ is a uniform. In each iteration, the attackers pushes it closer to the hard sample distribution m', while the defenders keep looking for the worst-case approximation m' within the uncertain set $B_\rho(m)$. Once the algorithm converges, i.e., $D(\tilde{m} \parallel m') \leq \epsilon$, we can ensure that the discrepancy is bounded above by a small value if $D(\cdot, \cdot)$ satisfies triangle inequality:

$$D(\tilde{m} \parallel m) \leq D(\tilde{m} \parallel m') + D(m' \parallel m) \leq \epsilon + \rho$$

If m' is exactly the worst distribution in the uncertain set $B_\rho(m)$, we can say that $D(\tilde{m} \parallel m) \leq |\epsilon - \rho|$, see Figure 5.1 for reference. As discussed in Chapter 3, Wasserstein metric is a true distance and satisfies the triangle inequality. Therefore, the learning task is completed when $\epsilon \to 0$, and the fake sample distribution $\tilde{m}$ will locate inside the Wasserstein ball of real data.

5.3 Learning Algorithm

In this section we provide learning procedure to solve the distributionally robust Nash equilibria under Wasserstein metric, and develop practical implementation algorithms to train deep generative models for image synthesis.

5.3.1 Learn the Distributional Robust Equilibrium

In DRG, the attacker and defender work against each other toward the robust Nash equilibrium by solving a minimax optimization problem in (4.5).

$$(P_j) : \inf_{a_j \in \mathcal{A}_j} \sup_{m' \in B_\rho(m)} \mathbb{E}_{\omega' \sim m'} l_j(a, \omega')$$

Since $B_\rho(m)$ is a subset of Lebesgue space (the set of integrable measurable functions under m), the original problem (P_j) has infinite dimensions, which does not facilitate the computation of robust optimal strategies. It has been presented in Chapter 4 that (P_j) can be reduced to a finite dimensional stochastic optimization problem when $\omega' \mapsto l_j(a, \omega')$ is upper semi-continuous and (Ω, d) is a Polish space. We

introduce a Lagrangian function for constraint (5.5),

$$\tilde{l_j}(a, \lambda, m, m') = \int_{\omega}^{'} l_j(a, \omega') dm' + \lambda(\rho - W(m, m')) \tag{5.6}$$

and the original problem (P_j) becomes

$$(\tilde{P_j}): \ \inf_{a_j \in \mathcal{A}_j, \lambda \geq 0} \sup_{m' \in B_\rho(m)} \tilde{l_j}(a, \lambda, m, m') \tag{5.7}$$

In robust game $\mathcal{G}(m)$, the defenders search for the worst-case hard sample distribution m' in the Wasserstein neighborhood of m to maximize its loss against the model $\tilde{m}$. According to the definition of Wasserstein metric with ground distance $d(\cdot, \cdot)$,

$$\begin{aligned} \sup_{m'} \tilde{l_j} &= \lambda\rho + \sup_{m'} \int_{\omega'} [l_j(a, \omega')] \ dm' - \lambda W(m, m') \\ &= \lambda\rho + \int_{\omega} \sup_{\omega'} [l_j(a, \omega') - \lambda d(\omega, \omega')] \ dm \end{aligned} \tag{5.8}$$

Define the integrand cost as

$$h_j(a, \lambda, \omega) = \lambda\rho + \sup_{\omega'} [l_j(a, \omega') - \lambda d(\omega, \omega')], \tag{5.9}$$

then $(\tilde{P_j})$ becomes a finite dimension problem on $\mathcal{A}_j \times \mathbb{R}_+ \times \Omega$ when $\mathcal{A}_j$ and Ω have finite dimensions

$$(\tilde{P_j^*}) \inf_{a_j \in \mathcal{A}_j, \lambda \geq 0} \mathbb{E}_m h_j(a, \lambda, \omega) \tag{5.10}$$

Since m is an unknown distribution observed by the noisy unsupervised dataset $\{x_1, \ldots, x_N\}$, it is challenging to compute the expected payoff $\mathbb{E}_m h_j(a, \lambda, \omega)$ and its partial derivatives. We need a stochastic learning algorithm to estimate the empirical gradients for the Wasserstein metric.

For a single player, the stochastic state ω_j leads to error

$$\epsilon_j = \nabla_{a,\lambda} h_j(a, \lambda, \omega_j) - \nabla_{a,\lambda} \mathbb{E}_m h_j(a, \lambda, \omega)$$

The variance of ϵ_j is high and not vanishing. To handle this, we introduce a swarm of players $\omega_j \sim m$, $j \in \mathcal{J}$, then the error term becomes

$$\epsilon = \frac{1}{|\mathcal{J}|} \sum_j \nabla_{a,\lambda} h_j(a, \lambda, \omega_j) - \nabla_{a,\lambda} \mathbb{E}_m h_j(a, \lambda, \omega)$$

It has zero mean and standard deviation as

$$\sqrt{\mathbb{E}[\epsilon^2]} = \frac{1}{|\mathcal{J}|} \sqrt{var[\nabla_{a,\lambda} h_j(a, \lambda, \cdot)]}$$

For realized $\omega \leftarrow \{x_1, \ldots, x_N\}$, the expected payoff is $\frac{1}{N} \sum_{j=1}^{N} h_j(a, \lambda, \omega_j)$, and the optimal strategy is

$$(a^*, \lambda^*) \in \arg\min_{a,\lambda} \sum_{j=1}^{N} h_j(a, \lambda, \omega_j)$$

This provides an accurate robust equilibrium payoff when N is very large.

5.3.2 A Toy Example

To illustrate the stochastic learning algorithm we consider specific robust games with finite number of players. Each player acts as if he is facing a group of opponents whose randomized control actions are limited to a Wasserstein ball, and tries to optimize the worst case payoff. The random variable ω is distributed over m and we assume it has finite p moments. We choose $|\mathcal{J}| = 2$, $p = 2$, $d(\omega, \omega') = \|\omega - \omega'\|_2^2$ and a convex payoff function $l_j(a, \omega')$ defined on $\mathbb{R}^2 \times \mathbb{R}^2$

$$l_j(a, \omega') = \|\omega' - a\|_2^2 = (\omega_1' - a_1)^2 + (\omega_2' - a_2)^2 \tag{5.11}$$

The optimal defender state ω'^* is computed through the Moreau-Yosida regularization, and the attacker's action pushes it closer to the destination ω as shown in Figure 5.3.

$$\begin{aligned} \sup_{m'} \tilde{l_j} &= \lambda\rho + \int_{\omega\in\Omega} \phi_j(a, \lambda, \omega) dm \\ \phi_j(a, \lambda, \omega) &= \sup_{\omega'\in\mathbb{R}^2} [l_j(a, \omega') - \lambda d(\omega, \omega')] \\ &= \sup_{\omega'\in\mathbb{R}^2} (\|\omega' - a\|_2^2 - \lambda\|\omega' - \omega\|_2^2) \end{aligned} \tag{5.12}$$

$$\omega'^* = \omega + \frac{\omega - a}{\lambda - 1}, \quad (\lambda > 1) \tag{5.13}$$

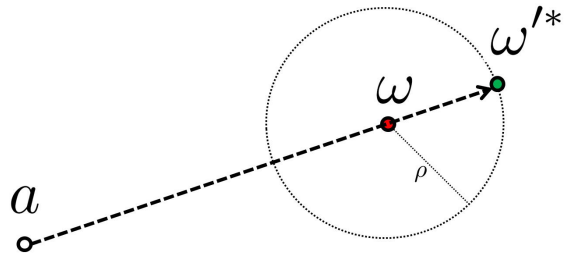

Figure 5.3: Action pushes the particle toward ω (which is unknown) given ω'^*

Then $d(\omega, \omega'^*) = \|\frac{\omega - a}{\lambda - 1}\|_2^2$, leads to the worst-case loss

$$l_j(a, \omega'^*) = \|\omega'^* - a\|_2^2 = \frac{\lambda^2}{(\lambda - 1)^2}\|\omega - a\|_2^2 \tag{5.14}$$

The Moreau-Yosida regularization on m' realized at ω'^* is

$$\begin{aligned} \phi_j(a, \lambda, \omega) &= l_j(a, \omega'^*) - \lambda d(\omega, \omega'^*) \\ &= \frac{\lambda}{\lambda - 1}\|\omega - a\|_2^2 \end{aligned} \tag{5.15}$$

The integrand cost function $h_j = \lambda\rho^2 + \frac{\lambda}{\lambda-1}\|\omega - a\|_2^2$. Thus, problem $(\tilde{P}_j^*)$ becomes

$$\inf_{a,\lambda} \mathbb{E}_m h_j = \inf_{a,\lambda} \int_\omega \lambda\rho^2 + \frac{\lambda}{\lambda-1}\|\omega - a\|_2^2 \, dm, \quad (\lambda > 1) \tag{5.16}$$

Given N observations, the stochastic robust loss is

$$\begin{aligned} l_N^* &= \frac{1}{N}\sum_{j=1}^{N} h_j(a, \lambda, \omega_j) \\ &= \lambda\rho^2 + \frac{\lambda}{N(\lambda-1)}\sum_{j=1}^{N}\|\omega_j - a\|_2^2 \end{aligned}$$

We set $\rho = 1$ and m is a dirac distribution where $\omega_j \equiv 1$. Figure 5.4 plots the trajectories of strategies during learning.

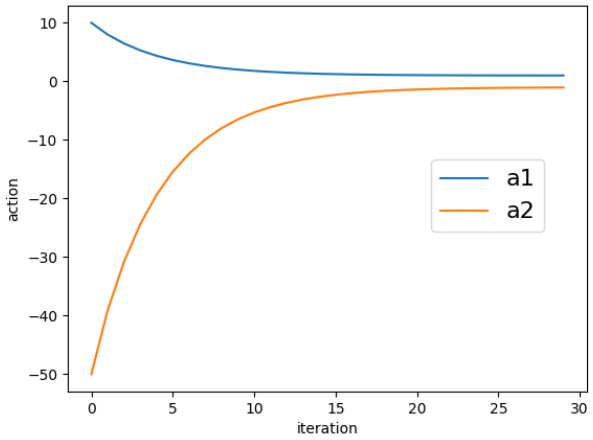

Figure 5.4: The optimal strategies converges to $(a_1^*, a_2^*) = (1, -1)$

5.3.3 Train Deep Generative Models

Generative models such as VAE [50], GAN [53], WGAN [103] have shown great success in recent years. VAE trains a encoder network and a decoder network by

minimizing the reconstruction loss, i.e., the negative log-likelihood with a regularizer. It tends to produce blurring images due to the additional noise terms in their model. GAN trains a generator network and a discriminator network by solving a minimax problem based on the KL-divergence. The model is unstable (see Figure 5.9) due to the discontinuity of the information-based loss functions, and the generator is vulnerable to saturation as the discriminator getting better. [102, 111] gives some empirical solutions to these problems, e.g., keeping balance in training generator and discriminator networks, designing a customized network structure. WGAN [103] defines a GAN model by an efficient approximation (Eq. 3.18) of the Earth Mover distance. During training, it simply crops all weights of the discriminator network to maintain the Lipschitz constraint.

In our framework, generative modeling is formulated as a distributionally robust game with two competitive groups of players, whose actions are defined on the parameter space $\theta = (\theta_a, \theta_d) \in \Theta$. In stochastic settings, ω, ω' and $\tilde{\omega}$ are instantiated to sample vectors $\{x_1, x_2, \ldots\}$, $\{x'_1, x'_2, \ldots\}$ and $\{\tilde{x}_1, \tilde{x}_2, \ldots\}$. The attacker produces indistinguishable artificial samples $\tilde{x}_i = G_{\theta_a}(z)$ to minimize the discrepancy $\inf_\theta D(\tilde{m}, m')$, where $\tilde{x}_i \sim \tilde{m}$. Meanwhile, the defender produce adversarial samples $x'_i = G_{\theta_d}(x_i)$, which are substitutes of the real ones, to maximize the loss $\sup_{m' \in B_\rho(m)} D(\tilde{m}, m')$, $x_i \sim m$, where $x'_i \sim m'$.

With Moreau-Yosida regularization, the defenders work on the following maximization problem to generate the optimal adversarial samples in Wasserstein ball $B_\rho(m)$,

$$\theta_d^* \in \arg\max_{\theta_d} \; l(\theta_a, \omega') - \lambda d(\omega, \omega')$$

and the attackers work on the minimization problem to find the best generative

Algorithm 3 DRGAN: Distributionally Robust Generative Model

Input: real data $(x_i)_{i=1}^N$, encoder network E, latent code $(\omega_i)_{i=1}^N$, batch size n, initial attacker parameters θ_{a0}, Lagrangian multiplier λ_0, initial defender parameters θ_{d0}, number of defender updates per attacker loop n_d, Wasserstein ball radius ρ, learning rate η, low-dimension random noise $z \sim \zeta$
Output: θ_a, θ_d, λ, fake data $\tilde{x}_i$
while θ_a has not converged **do**
  for $t = 1, 2, \ldots, n_d$ **do**
    Sample $(x_i)_{i=1}^n \sim m$ from real dataset
    Sample $(\tilde{x}_i)_{i=1}^n \sim \tilde{m}$ from generator $G_{\theta_a}(z)$
    $\omega_i \leftarrow E(x_i), \quad \tilde{\omega}_i \leftarrow E(\tilde{x}_i)$
    Perturb to generate hard samples $\omega_i' \leftarrow G_{\theta_d}(\omega_i)$
    $g_d \leftarrow \nabla_{\theta_d}\, l(\tilde{\omega}_1^n, {\omega'}_1^n) - \lambda d(\omega_1^n, {\omega'}_1^n)$
    $\theta_d \leftarrow \theta_d + \eta RMSProp(g_d)$
  end for
  Sample $(x_i)_{i=1}^n \sim m$ from real dataset
  Sample $(\tilde{x}_i)_{i=1}^n \sim \tilde{m}$ from generator $G_{\theta_a}(z)$
  $\omega_i \leftarrow E(x_i), \quad \tilde{\omega}_i \leftarrow E(\tilde{x}_i)$
  Perturb to generate hard samples $\omega_i' \leftarrow G_{\theta_d}(\omega_i)$
  $g_{a,\lambda} \leftarrow \nabla_{\theta_a,\lambda}\, \lambda\rho + l(\tilde{\omega}_1^n, {\omega'}_1^n) - \lambda d(\omega_1^n, {\omega'}_1^n)$
  $\theta_a \leftarrow \theta_a - \eta RMSProp(g_{a,\lambda})$
  $\lambda \leftarrow \lambda - \eta RMSProp(g_{a,\lambda})$
end while

parameters θ_a^*

$$\theta_a^* \in \arg\min_{\theta_a,\lambda}\ \lambda\rho + l(\tilde{x}, x'^*) - \lambda d(x, x'^*)$$

Given enough observations $\{x_1, x_2, \ldots\}$ from the unknown real distribution m, a similar distribution $\tilde{m}$ can be learned by solving the distributionally robust Nash equilibrium. New samples generated from $\tilde{x}_i \sim \tilde{m}$ should be indistinguishable from the real ones. The generative model based on DRG is summarized in Algorithm 3.

5.4 Experiments

We evaluate the proposed approach on both synthetic and real datasets. Several tasks will be investigated: (i) unsupervised learning for clustering. (ii) shallow generative networks on toy datasets. (iii) deep generative networks for image synthesis.

5.4.1 Unsupervised Learning for Clustering

We apply our algorithm to learn a three-dimensional data distribution for the Fisher's Iris dataset [112]. This dataset contains 150 samples, each having five attributes: petal length, petal width, sepal length, sepal width and class. We remove the class label and using PCA to extract three most prominent features. The observed data samples are plotted in Figure 5.6, each color represents a class of flower. The generative model is set as the affine transformations of three unit balls z_1, z_2, z_3. Initially, the attacker parameters W_1, W_2, W_3 are set as identity matrices and $b_1 = b_2 = b_3 = 0$, so all fake samples are located in a unit ball shown in Figure 5.6.

$$\tilde{x} = G_{\theta_a}(z) \in \{W_1 z_1 + b_1, W_2 z_2 + b_2, W_3 z_3 + b_3\}$$

$$x' = G_{\theta_d}(x) = W_0 x + b_0$$

Hard samples are generated by perturbing the real ones using another affine transformation with parameters W_0, b_0. Since the dimension is low, we directly work with the raw data vectors. We follow algorithm 3 and set $n_d = 20, \rho = 0.1, \lambda_0 = 10, \eta_a = 0.1, \eta_d = 0.01$. The training cost for attackers and defenders at each iteration are displayed in Figure 5.5. In each defender loop, the hard samples are updated to maximize the Wasserstein loss $W(\tilde{m}, m')$, and then the attacker

refines the generative model to optimize its worst-case performance. After 150 iterations, the algorithm converged at the distributionally robust Nash equilibrium. The generated samples (black dots in Figure 5.6) successfully covered the region of real samples (color dots), which demonstrated the effectiveness of our algorithm.

Next section will show the application of DRG on image synthesis, in which the attackers and defenders are realized with deep neural networks.

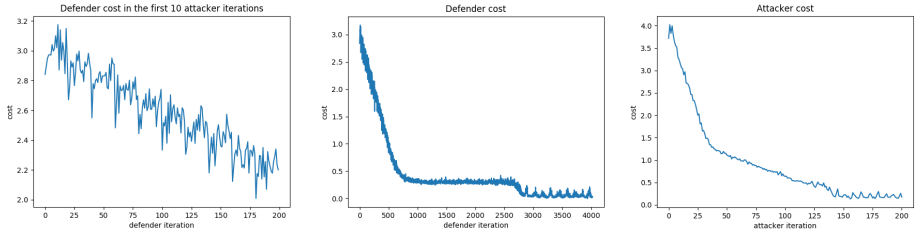

Figure 5.5: Attacker and defender cost during training

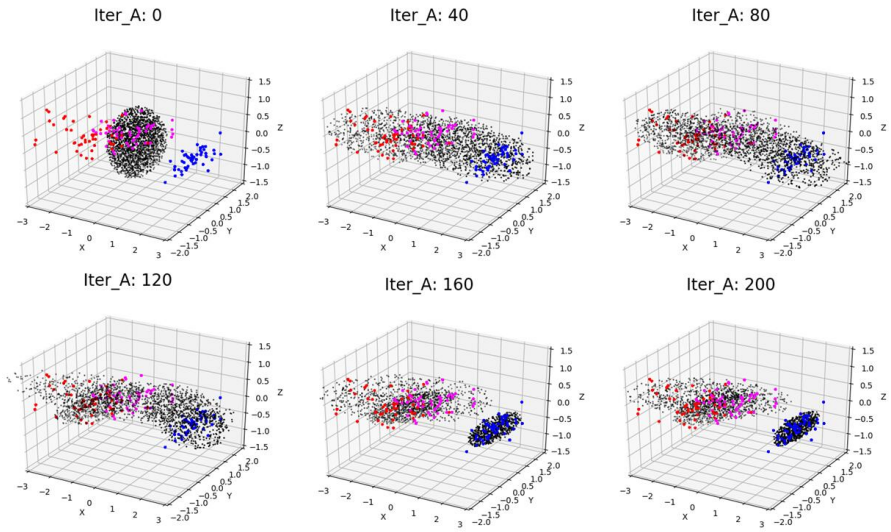

Figure 5.6: Generative modeling for Iris Flower dataset

5.4.2 Shallow Generative Networks on Toy Datasets

We conduct simulations on three toy datasets: Swiss Roll, 8Gaussians and 25Gaussians. The experiment settings are same as in [113, 114] for fair comparison. Our model perturbs the sample points before sending to the discriminator and generator. The perturbation is calculated by maximizing the Wasserstein distance between real and fake distributions. We borrow the idea from adversarial training and use the Fast Gradient Sign Method (FGSM) [115] to perturb the samples. FGSM is an attack bounded by Chebyshev distance ($l_\infty(x, x')$). We replace the loss function by Wasserstein distance ($W(m, m'), x \sim m, x' \sim m'$) to bound the perturbation within a Wasserstein ball. The perturbed example is computed as

$$x' = x + \rho \cdot sgn(\nabla_x W(x_1^n, \tilde{x}_1^n))$$

where n is the batch size, and we set $n = 1024$. This is a simple one-step maximization of the inner loop in our DRG framework (Algorithm 3). There are many other powerful multi-step perturbation schemes [116] to solve the inner optimization problem.

Since all data points are in 2D space, we can compute the exact Wasserstein distance to evaluate the model. In addition, we report Sinkhorn distance (an approximated version of Wasserstein distance) [108] and Fréchet inception distance (FID) [63]. We repeated the simulation by 1000 times. The visual and numerical results are displayed in Figure 5.7 and Table 5.1. Learning curves are illustrated in Figure 5.8. Evaluation results show the superiority of our models in both sample diversity and convergence rate.

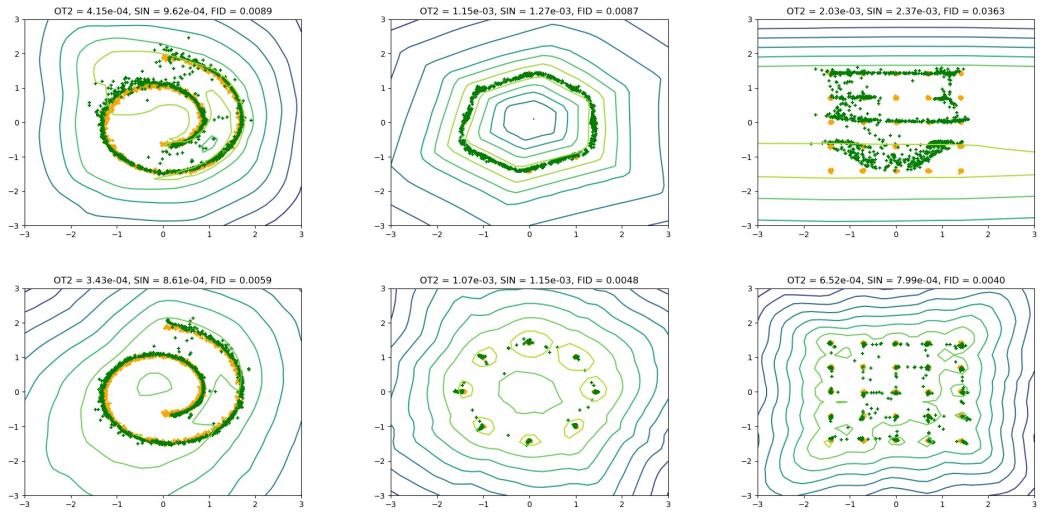

Figure 5.7: Visual results for generated samples on toy datasets. orange dots: real samples, green dots: generated samples, curves: discriminator contours. First column: swissroll, second column: 8gaussians, third column: 25gaussians. Top: WGAN. Bottom: DRGAN.

Table 5.1: Numerical results for generated samples on toy datasets. First block: swissroll, second block: 8gaussians, third block: 25gaussians. In each block: second row is WGAN and third row is DRGAN.

	Exact Wasserstein	Sinkhorn Distance	FID
real-real	$3.00e-04 \pm 1.23e-04$	$8.27e-04 \pm 1.33e-04$	0.0045 ± 0.0034
real-fake	$3.99e-04 \pm 1.29e-04$	$9.31e-04 \pm 1.36e-04$	0.0082 ± 0.0046
real-fake	$3.51e-04 \pm 1.43e-04$	$8.68e-04 \pm 1.51e-04$	0.0067 ± 0.0047
real-real	$1.17e-03 \pm 4.44e-04$	$1.25e-03 \pm 4.74e-04$	0.0050 ± 0.0042
real-fake	$1.18e-03 \pm 2.36e-04$	$1.31e-03 \pm 2.49e-04$	0.0059 ± 0.0040
real-fake	$1.17e-03 \pm 4.36e-04$	$1.24e-03 \pm 4.60e-04$	0.0059 ± 0.0044
real-real	0.0039 ± 0.0009	0.0039 ± 0.0009	0.0055 ± 0.0035
real-fake	$1.96e-03 \pm 1.87e-04$	$2.31e-03 \pm 1.94e-04$	0.0332 ± 0.0111
real-fake	$8.26e-04 \pm 2.17e-04$	$9.92e-04 \pm 2.34e-04$	0.0056 ± 0.0042

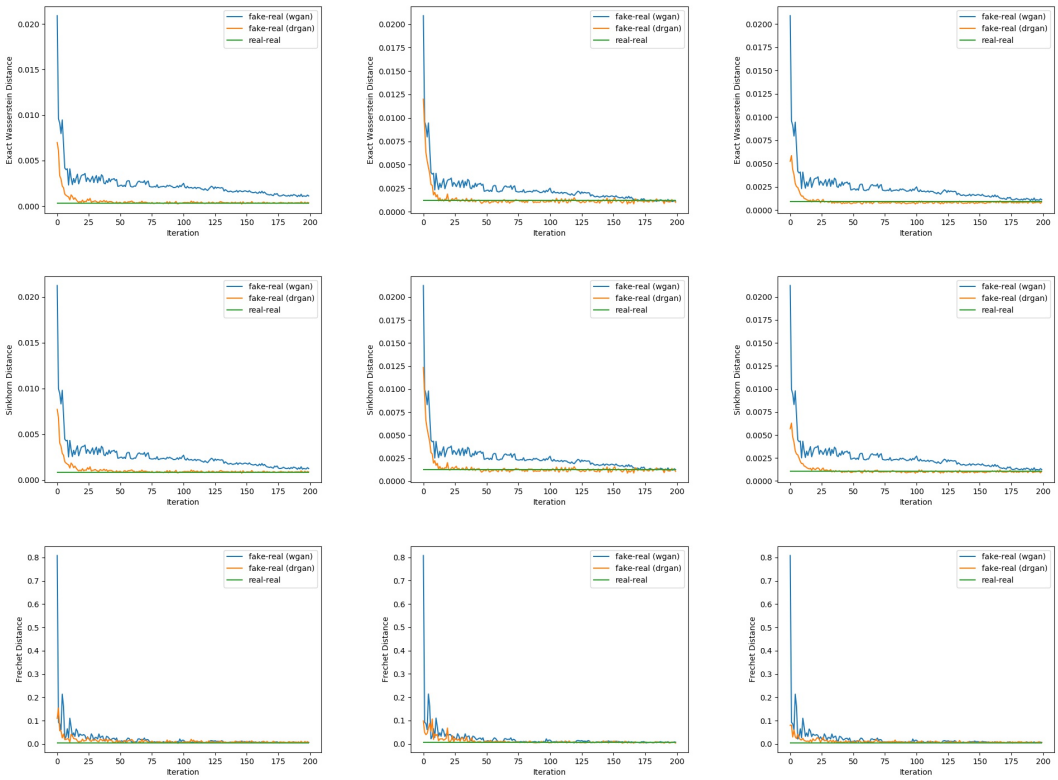

Figure 5.8: Learning curves on toy datasets. Orange line: DRGAN, blue line: WGAN, green line: value between two real distributions. First column: swissroll, second column: 8gaussians, third column: 25gaussians. First row: exact Wasserstein distance (OT2), second row: Sinkhorn distance (SIN), third row: Fréchet inception distance (FID)

5.4.3 Deep Generative Networks for Image Synthesis

We apply DRGAN on two popular benchmarks: CIFAR-10 [117] and CelebA [98]. The objective is to generate fake images with lifelike objects such as ship, cat, and human face. We compare DRGAN with unsupervised generative models DCGAN [95], WGAN [103], WGAN-GP [113], and Rob-GAN [118].

The CIFAR-10 dataset consists of 60K 32×32 colour images equally split in 10 classes. We follow the experimental settings of WGAN [103] that only uses the 50K training samples for a fair comparison. Each data point x_i has 3072 dimensions. The CelebA dataset consists of 202K 64×64 cropped human face images. Each data point x_i has 12288 dimensions. Fake samples are generated from low-dimensional noise vectors $z \sim \zeta$, where ζ is a random normal distribution with 128 dimensions.

Network Structure In the experiment, the generator network $x = G_{\theta_a}(z)$ takes the same architecture as in DCGAN [95]. The real samples are modified before fitting into the discriminator. We use FGSM to generate hard samples x' and the perturbations are constrained within a Wasserstein ball. For the critic network $y = E_{\theta_d}(x)$, we use 1 CNN-ReLU layer followed by 3 CNN-BatchNorm layers and a fully connected layer to produce code vectors. The closest work to us is Rob-GAN [118], in which an adversarial attack was added to GAN training. The adversarial attacker perturbs real data toward the fake distribution to train a robust discriminator. In our approach, the real samples are perturbed away from the fake distribution to train a robust generator. Moreover, we use Wasserstein metric as the loss function. It is a true distance and therefore the triangle inequality ensures the error to be bounded when the algorithm converges, as shown in Figure 5.1. In addition, DRGAN is an unsupervised model which does not require the knowledge

of class labels.

Loss Functions In our model, the Wasserstein distance $l(\tilde{x}^n, x'^n)$ is implemented by Sinkhorn-Knopp's algorithm [119], and the ground cost $d(x^n, x'^n) = \frac{1}{n}\sum_{i=1}^{n} \|x_i - x'_i\|_2^2$. Instead of directly computing the L2-norm on raw data vectors, algorithm 3 uses an encoder network $E_{\theta_d}(x)$ to learn feature codes and then fit it to the critic. GAN models are hard to train due to the unstable gradients. Some methods suggest a lower learning rate, and others use weight clipping [103] or gradient penalty [113] to regularize the discriminator.

We will show that training a robust generator has the same effect of enforcing a Lipschitz constraint on discriminator or reducing the learning rate. Suppose a robust generator (attacker) can successfully fool the discriminator (defender) at iteration k, then it cannot be defeated by the defender in few training steps. Let $\delta_D \geq 0$ be the increase of the defender's reward from k to $k+1$, and w_k be the network weights at iteration k. For a fixed generator, we have

$$\begin{aligned}\delta_d &= D(G; w_k) - D(G; w_{k+1}) \\ &\approx \nabla D(G; w_k)\|w_k - w_{k+1}\| \\ &\leq C_d\|w_k - w_{k+1}\|\end{aligned}$$

where C_d is the Lipschitz constant of D. As we can see, either decreasing the learning rate on w or setting a lower Lipschitz constant on D leads to a smaller δ_d, which indicates a more robust generator. On the other hand, for a fixed

discriminator

$$\begin{aligned}\delta_d &= D(G(z;w_k)) - D(G(z;w_{k+1})) \\ &\approx \nabla D(G(z;w_k))[G(z;w_k) - G(z;w_{k+1})] \\ &\approx \nabla D(G(z;w_k))\nabla G(z;w_k)\|w_k - w_{k+1}\| \\ &\leq C_d C_g \|w_k - w_{k+1}\|\end{aligned}$$

where C_g is the Lipschitz constant of G. For a large C_g, i.e. the generator is not robust, the discriminator can easily get a large reward δ_d with just a small move. In that case, the discriminator will be too strong and cause vanishing gradients. Our approach directly trains a robust generator with perturbed samples instead of enforcing the Lipschitz constraint as in [103], which might be too conservative.

Hyperparameters The encoder maps the original data into a 100-dimension feature space, which matches the dimension of the random noise z. In all experiments, the cost based on Wasserstein metric is normalized to $[0, 1]$, where the supremum indicates the cost between images that are all black and all white. The hyperparameters listed in Algorithm 3 are chosen by validation and listed in table 5.2; others are set as the default values in their references. For training we choose the RMSProp optimizer [120] because it doesn't involve a momentum term. Empirically, we found momentum-related optimizers may deteriorate the training. The reason is, in robust games the payoff function is dynamic and changes every time the other players take actions. Since the structure of the objective surface is not stationary, it's meaningless to follow the velocity of the previous optimization steps.

Table 5.2: Hyper parameters

parameters	n	ρ	λ_0	n_d	η	$\theta_{a0}, \theta_{d0}, \eta$
values	64	0.1	10	1	0.00005	random normal

Evaluation We use two widely used metrics to measure the quality of generated images: Inception Score (IS) and Fréchet Inception Distance (FID). The former one fit generated samples to an Inception-v3 network pretrained on ImageNet, and compute the KL divergence between the conditional label distribution and its marginal distribution. Large value indicates images with meaningful objects and high diversity.

$$IS = \exp(\mathbb{E}_{x\sim\mathbb{P}_g} D_{KL}(P(y \mid x) \parallel P(y))) \tag{5.17}$$

The FID compares the statistics of generated samples to real ones, by calculating the Fréchet distance between two multivariate Gaussians. For images, the data is encoded by an Inception-v3 network. The codes for comparison have 2048 dimensions and are approximated by Gaussians: $x_r \sim \mathcal{N}(\mu_r, \Sigma_r)$, $x_g \sim \mathcal{N}(\mu_g, \Sigma_g)$.

$$FID = \|\mu_r - \mu_g\|^2 + Tr(\Sigma_r + \Sigma_g - 2(\Sigma_r\Sigma_g)^{\frac{1}{2}}) \tag{5.18}$$

Numerical evaluation results on CIFAR-10 are listed in Table 5.3. The first row evaluates the quality of real images as a upper limit for all generative models. Our model significantly outperforms WGAN [103] because weight clipping is a very simple approximation of Wasserstein distance and it ignores higher moments of the data distribution. Our model performs comparably to WGAN-GP [113] because training a robust generator has the same effect of enforcing a Lipschitz constraint by gradient penalty as mentioned above. Rob-GAN [118] combines the ideas of

adversarial training and GAN training, but performs not well in image synthesis. Because it was designed for robust classification with GAN data augmentation rather than learning generative models.

Table 5.3: Numerical results for generated images on CIFAR-10

	FID	IS
real-real	2.07	9.67032 ± 0.08135
real-fake (dcgan)	20.32	4.95168 ± 0.04404
real-fake (wgan)	47.51	3.34858 ± 0.03147
real-fake (wgan-gp)	19.71	4.75546 ± 0.04454
real-fake (rob-gan)	38.28	4.35532 ± 0.03620
real-fake (drgan)	18.76	5.16395 ± 0.04505
real-fake (swgan)	17.81	5.11680 ± 0.04115

Visual results on CelebA dataset are shown in Figure 5.9, in which the last row lists the most similar samples in the real dataset. The training curve for DRGAN is plotted in Figure 5.11. It means the Wasserstein loss is highly related to the sample quality. By optimizing the worst-case loss function, the DRGAN model converges very quickly to the real data distribution and successfully produce sharp and meaningful images. In experiments we found that the original GAN generator [95] suffers from unpredictable quality deterioration at iteration $5.3K$, $7.8K$, $10.2K$ (Figure 5.10), etc, while our algorithm keeps improving the sample quality. This problem is caused by the discontinuity of the KL-divergence.

The evaluation of generative models is itself a research topic. [121] figured out that different evaluation metrics favor different models. For example, a high log-likelihood doesn't mean good visual quality, and vice versa. Therefore, the metric used in training and evaluation should fit for the specific application. In our case, the learned fake data distribution should be as close as to the real one.

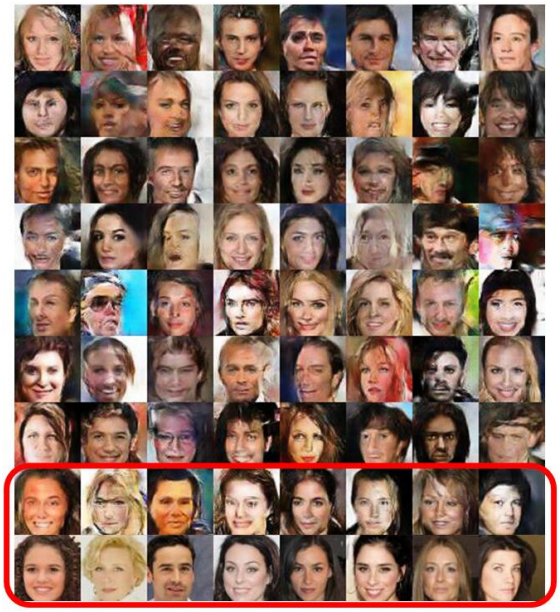

Figure 5.9: DRGAN results on CelebA, attacker iteration at 300K

So we directly measure the Wasserstein distance between the two distributions, though it's very time consuming. We compare our algorithm with DCGAN [95] and WGAN [103], and report the quantitative results in Table 5.4.

The computation complexity per attacker iteration is linear $O(n)$ with respect to the batch size. We use a Titan Xp to train the model and plot the computation time in Figure 5.12. When $n = 64$, it takes 0.2 seconds for an attacker update. Our algorithm has smaller constant factor than WGAN.

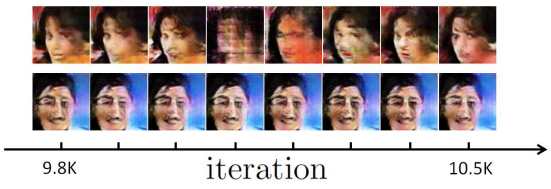

Figure 5.10: Stability of the generated models. Upper: DCGAN, Bottom: DRGAN

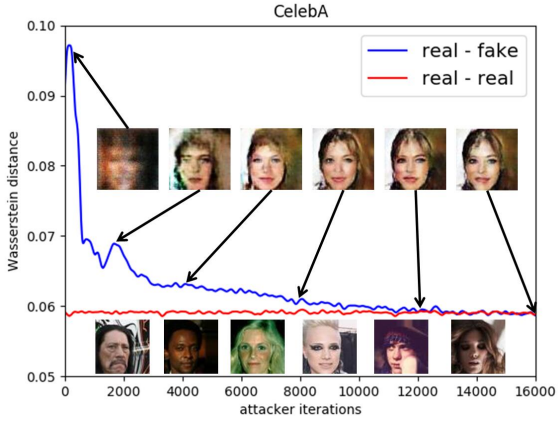

Figure 5.11: Training curve of DRGAN using Wasserstein metric. The loss goes down as generated samples getting better, and converges to the Wasserstein distance between two real data sets. The curves are smoothed for visualization purpose.

Table 5.4: Performance evaluation

$W(m, \tilde{m})$ $(\times 10^{-5})$	1K samples	10K samples
real - real	12.9	1.74
real - DRGAN	**22.6**	**15.9**
real - DCGAN	37.3	16.4
real - WGAN	31.0	17.2

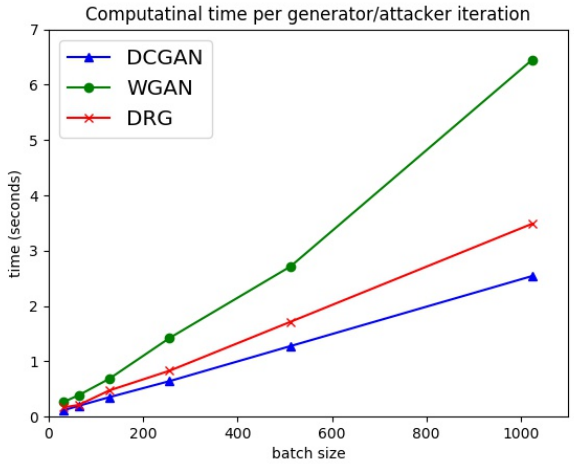

Figure 5.12: Computation time with respect to batch size

5.5 Discussion

We proposed a new game theory model based on Wasserstein distance to train generative models. In this game, two competing groups of players work on a robust optimization problem to reduce the discrepancy between model and data. The defenders compute perturbations of real data to produce a set of hard examples that has the maximum distance from the model distribution, while the attackers are trained on the hard sample set to push the model mass toward the manifold of real data. Instead of prevalent information-based loss functions such as KL-divergence, we use Wasserstein distance to measure the similarity between distributions. Its advantages have been analyzed from both theoretical and empirical perspectives. We developed a learning procedure to solve the robust Nash equilibrium, and offered practical realizations on deep neural networks. The approach has been applied to image synthesis and tested on real datasets. It can successfully produce artificial samples with good visual quality and high diversity. The learning process is stable and converges fast. Experiment evaluation shows our algorithm achieving significantly better performance than DCGAN and WGAN in terms of IS, FID, and the Wasserstein distance between the real and fake data distributions.

To our knowledge, this is the first work connecting distributionally robust game with deep generative modeling. It would be interesting to apply similar techniques to generate sequential data such as speech and video. Another direction is to study the properties of Wasserstein space and develop more efficient algorithms for robust optimization. Next chapter will discuss another kind of generative modeling technique: style transfer, and its application in voice conversion.

Chapter 6

Emotional Voice Conversion: A Style Transfer Approach

The original objective of generative modeling is to generate new samples from scratch by learning complicated data distributions. Similar techniques can be extended to conditional generative models that manipulate the existing data and add user-specified properties. In this chapter, we first study the problem of style transfer that maps data from one domain to another, and then work on a specific problem in voice conversion. In the mapping, domain-specific information (i.e., the *style*) is discarded and manipulated, while domain-invariant knowledge (i.e., the *content*) is preserved. The disentangled representations of style and content are recognized by domain classifiers and modeled by autoencoders. We propose a game-theoretic approach to learn the mapping. The encoders, decoders and classifiers form a distributionally robust game with competitive and collaborative agent coalitions.

In the second part of this chapter, we develop a nonparallel data-driven emo-

tional speech conversion algorithm. It enables the transfer of emotion-related characteristics in the speech signal while preserving the speaker's identity and the linguistic content. Most existing approaches require parallel data and time alignment, and they are not applicable in many real situations. We achieve nonparallel training based on an unsupervised style transfer technique that learns a translation model between two distributions instead of a deterministic one-to-one mapping between paired examples. The conversion model consists of an encoder and a decoder for each emotion domain. We assume that the speech signal can be decomposed into an emotion-invariant content code and an emotion-related style code in the latent space. Emotion conversion is performed by extracting and recombining the content code of the source speech and the style code of the target emotion. We test the proposed approach on a large nonparallel corpora with thousands of utterances, and the model is able to transfer between four emotions: angry, happy, neutral, sad. This chapter is based on the INTERSPEECH 2019 paper "Nonparallel Emotional Speech Conversion" [122] which was done in collaboration with Deep Chakraborty, Hamidou Tembine, and Olaitan Olaleye.

6.1 Introduction

Style transfer originally means rendering an image in the style of other images. This meaning can be extended to other kinds of data like music and speech. More generally, it refers to mapping data from one domain to anther while keeping its semantic content or domain-invariant knowledge. For example, transfer photographs to artistic paintings, convert one person's voice to another, or translate music to imitate different instruments. Another case is transfer learning. It adapts a pre-

trained model in source domain to classify samples in target domain where labeled data is limited.

Let $x_1 \in \mathbb{X}_1$ be samples in the source domain and $x_2 \in \mathbb{X}_2$ be samples in the target domain. The goal of style transfer is to learn a mapping function $\mathcal{T} : \mathbb{X}_1 \to \mathbb{X}_2$ such that the generated output $x'_{2\leftarrow 1} = \mathcal{T}(x_1)$ is indistinguishable from the real samples drawn from the target domain. The optimal mapping $\mathcal{T}^*$ transforms x_1 to $x'_{2\leftarrow 1}$ such that they are distributionally equivalent $x'_{2\leftarrow 1} \overset{d}{=} x_2$. Semantic content should be preserved during the transformation.

We propose a game model to learn the transform. The domain-invariant content information and domain-specific style information are decomposed by disentangled representation learning. For high dimensional data like image and speech waveform, we employ autoencoders to independently model the high-level semantic content and the low-level style information. The learning problem is formulated as a distributionally robust game with cooperative agents and payoff uncertainty. In this game, several groups of players run with different objectives. The intergroup competition and intragroup collaboration enable the players to learn from each other and optimize their worst-case performance.

This work has a wide range of applications. In visual and performing arts, it's inspiring to automatically generate artificial paintings with user-specified style or play synthetic music with desired timbre and musical instrument. In informatics, it's useful to transform speaker identity by modifying his voice to sound like another person. It is also possible to learn and mimic animal's vocalization and study the feedback on the artificially generated sound. For case study, we apply our approach in emotional voice conversion.

6.2 Related Work

There are several topics closely related to this work. We will discuss their connections and differences to our task.

6.2.1 Deep Generative Modeling

Different from discriminative models, this approach models the full distribution of data instead of only the target variables. It supports unsupervised training. At test time, it takes random noise as input and outputs realistic samples. This model can extend to conditional generative models that take in additional information and produce user-specified outputs. Generative adversarial networks (GANs) [53], variational auto-encoders (VAEs) [50], and auto-regressive models [74] are the three main-stream approaches.

6.2.2 Image Style Transfer

There are two types of style transfer problems: example-based style transfer where the style comes from one image, and domain-based style transfer where the style is learnt from a collection of images in a specific domain. The former problem originates from non-photorealistic rendering (NPR) [123] in computer graphic, and has the similar meaning of realistic image manipulation. The goal is to edit image in a user-specified way and keep it as realistic as possible. Practical issues include texture synthesis and transfer [124], photo manipulation of shape and color [125], photorealistic image stylization [126], etc. In general, the output should be similar to the input in high-level structures and varies in low-level details such as color and texture.

Recently, Gatys et al. [127] claimed the image content and style information are separable in Convolutional Neural Network representations. They introduced a method [80] to separate and recombine content and style of natural images by matching feature correlations (Gram matrix) in different convolutional layers. However, their synthesis process is slow (an hour for a 512*512 image). Moreover, the style from a single image is ambiguous and may not capture the general theme of an entire domain of images.

The second problem, also known as image-to-image translation, learns a mapping to transfer images from one domain to another. For example, super-resolution [69] maps low-dimensional images to high-dimension, colorization [128] maps gray images to colorful images; other cases include day to night, dog to cat, young to old, summer to winter, photographs to paintings, aerial photos to satellite maps [126, 129, 130, 131, 132, 133, 134]. The mapping can be learnt in a supervised or unsupervised manner. In supervised settings [71, 135, 136], corresponding image pairs across domains are available for training. In unsupervised settings [129, 137, 138, 139, 140], there's no paired data and the training set only contains independent set of images for each domain. Our work is under the unsupervised setting because it is more applicable, and the training data is almost free and unlimited.

6.2.3 Domain Adaptation

Most recognition algorithms are developed for and evaluated on a limited number of public datasets like ImageNet, MS-COCO, CIFAR-10, and MNIST. In real applications, these algorithms often confront performance degradation when applied to a new domain.

In unsupervised domain adaptation, source domain has labeled data $\mathcal{D}_s = \{x_i^s, y_i^s\}_{i=1}^{n_s}$ while target domain contains data without labels $\mathcal{D}_t = \{x_i^t\}_{i=1}^{n_t}$. The goal is to learn a classifier $f : x_i^t \mapsto y_i^t$ for the unseen target samples by exploring the knowledge learnt from the source domain. Domain adaptation algorithms attempt to transfer knowledge across domains by solving the domain shift problem, i.e., the data-label distributions $p(x^s, y^s)$ and $p(x^t, y^t)$ are different.

There are many approaches to address this issue. One is to extract transferable features that are invariant across domains [141, 142], or learn representative hash codes [143] to find a common latent space where the classifier can be used without considering the data's origin. Another trend is to learn the transformation between domains [144] to align the source and target data points through barycentric mapping, and train a classifier on the transferred source data. Courty [145] and Damodaran [146] proposed to look for a transformation that matches the data-label joint distributions $p(x^s, y^s)$ in source domain to its equivalent version $p(x^t, y^t)$ in target domain. The predictive function f is learnt by minimizing the optimal transport loss from $p(x^s, y^s)$ to $p(x^t, f(x^t))$. As a by-product, minimizing the optimal transport cost is equivalent to mapping a source domain sample to a target domain sample with similar semantic content, and this is the domain transfer problem.

6.2.4 Voice Conversion

Voice conversion (VC) aims to change a speaker's voice to make it sounds like spoken by another person. It is a special case of voice transformation (VT), whose goal is to modify human speech without changing its meaning. VC transforms speaker identity by replacing speaker-dependent components of the signal while

maintaining the linguistic information. Speech quality and speaker similarity are two important factors to evaluate a VC system. There are a bunch of VC applications, such as movie dubbing, personalized TTS (Text To Speech) systems, speaker accent or emotion transformation, speaking-aid devices, call quality enhancement, etc.

Most VC frameworks involve three steps: feature extraction, feature conversion, and waveform generation. In speech analysis, waveform signals are encoded into feature representations that are easy to control and modify. Spectral envelope, mel-cepstrum, fundamental frequency (f_0), formant frequencies and bandwidths are the most widely used features to represent speech in short-time segments. To capture contextual information across frames, implicit methods such as hidden Markov models (HMMs), Long Short-Term Memory (LSTM) and recurrent neural networks (RNNs) [147] were developed.

The main work in VC is to transform the source feature sequences to target feature sequences that capture the speaker identity. Most traditional VC systems perform frame-by-frame mapping under the assumption that speech segments are independent from each other. Some recent models such as HMM and RNN incorporate speech dynamics implicitly. There are four typical approaches to learn the mapping function: codebook mapping (e.g., Vector quantization (VQ) [148]), mixed linear mappings (e.g., Gaussian mixture model (GMM) [149]), neural network mapping (e.g., RBM, DNN, RNN [150]), and exemplar-based mapping (e.g., non-negative matrix factorization (NMF) [151]). Beyond these, an autoregressive neural network model called WaveNet [75] was proposed. It can directly learn the mapping based on raw audio and generate speech waveforms conditioning on the speaker identity.

There are various assumptions in speech analysis and waveform generation.

Source-filter models assume speech to be generated by excitation signals passing through a vocal tract, and encode speech waveforms as acoustic features that represent sound source and vocal tract independently. However, the original phase information will lose under this assumption. At conversion time, the converted target features are passed through a vocoder based on the source filter model to reconstruct the waveform. Quality degradation may happen due to the inaccurate assumption. Iterative phase reconstruction algorithm Griffin-Lim [152] was adopted to alleviate this issue. Harmonic plus noise models (HNM) [153] assume speech to be a combination of a noise component and a harmonic component, i.e., sinusoidal waves with frequencies relevant to pitch. Speech is parameterized by the fundamental frequency f_0 and a spectrum which consists of a lower band of harmonic and a higher band of noise. Other assumptions include stationary speech signal, frame-by-frame mapping, time-invariant linear filter, etc. Recently, Tamamori et. al [154] proposed a speaker-dependent WaveNet vocoder that does not require explicit modeling of excitation signals and those assumptions.

In terms of conversion conditions, VC can be categorized into parallel and non-parallel, text-dependent and text-independent systems [155]. In parallel systems, the training corpus consists of paired recordings from the source and target speakers with same linguistic contents. The shared acoustic features can be used to train the mapping model. To get parallel feature sequences of equal length, a time-alignment step must be included to remove the temporal differences in the recordings, for example, the dynamic time warping (DTW) algorithm [148]. Phoneme transcriptions are also useful for time alignment. Non-parallel system does not require sentences with the same linguistic contents. It is much more useful and practical because non-parallel speech data is easier to collect and therefore

can get larger training sets. There are several ways to learn the mapping without paired data: (1) use unit selection [156] to choose matched linguistic feature pairs; (2) build pseudo parallel sentences on extra automatic speech recognition (ASR) modules [157]; (3) extract speaker-independent features in shared latent space [158]; (4) use unpaired image-to-image translation approaches [71, 139, 140].

Parallel, text-dependent systems are supposed to have better performance. However, parallel utterance pairs are difficult to get. Most parallel VC systems require time alignment to extract parallel source-target features. The misalignment in automatic time alignment algorithms often leads to degradation in speech quality, while manual correction is arduous. Recently, the winner of VC Challenge 2018 [159] showed their algorithm can achieve similar results in both parallel and non-parallel settings. It first uses a lot of external speech data with phonetic transcriptions to train a speaker-independent content-posterior-feature extractor, followed by a speaker-dependent LSTM-RNN to predict fundamental frequency f_0 and STRAIGHT spectral features [113], and then reconstruct the waveforms with a speaker-dependent WaveNet vocoder [154]. Moreover, Kaneko et.al [160] and Fang et. al [153] claimed their nonparallel, text-independent VC algorithms based on CycleGAN [138] perform comparable to or better than the state-of-the-art parallel approaches.

6.3 Motivation

In machine learning, discriminative models predict labels from data by learning a conditional distribution $p(y \mid x)$, while generative models synthesize new data with desired labels by drawing samples from estimated distribution $p(x \mid y)$.

From a perspective of probabilistic modelling, style transfer learns two conditional distributions $p(x_1 \mid x_2)$ and $p(x_2 \mid x_1)$. When paired data is available, it is easy to infer from the joint distribution $p(x_1, x_2)$. For nonparallel data, the problem is ill-posed because the joint solution is not unique given two marginal distributions $p(x_1), p(x_2)$.

To solve this problem, additional constraints are required. Some researchers proposed to keep a particular part of the data unchanged, e.g., pixel intensity, gradient or object boundaries [137, 161]; others suggested to preserve some properties of the data, such as semantic features or class labels [129].

Zhu et al. proposed a very straightforward constraint called cycle-consistency [138]. It assumes that if a sample is translated from source domain to target domain and then translated back, it should be unchanged. Choi et al. [162] generalized it to perform multiple-domain translation using a single generative model. However, domain transfer is not a one-to-one mapping, but many-to-many. In some cases, the cycle-consistency constraint is too strong to provide enough diversity in the translated outputs.

Based on a similar idea, Liu et al. [139] developed the UNIT framework by making a fully shared latent space assumption, in which corresponding images across domains can be mapped to a same latent code in shared-latent space. This assumption implies the cycle-consistency constraint. Xun et al. [140] extended it to a partially shared latent space assumption, where each example is generated from a shared content code and a domain-specific style code. Images are translated across domains by replacing the style code.

Some approaches [141, 162, 163] assume there exists a transformation $\mathcal{T}$ such that the source and target data can be matched in a new representation $p(\mathcal{T}(x_1)) =$

$p(\mathcal{T}(x_2))$. The types of transformation includes projections, affine transform, and non-linear mapping defined by neural networks. The objective is to minimize the gap between two transformed distributions. Several metrics used to compare distributions have been discussed in chapter 3. One requirement is that $p(x_1)$ and $p(x_2)$ share a common support, otherwise the optimization process is instable due to vanishing gradient issues.

Another idea is to directly assume the mapping $\mathcal{T} : x_2 = \mathcal{T}(x_1)$ to be the optimal transport between $p(x_1)$ and $p(x_2)$. The optimal solution $\mathcal{T}^*$ minimizes the global transportation cost (i.e., Wasserstein distance) between the source and target distributions. As a by-product, minimizing the transport cost gives the alignment of corresponding samples as well as their labels. It can be used to solve the domain adaptation problem [146] by finding the joint distribution optimal transport between $p(x_1, y_1)$ and $p(x_2, f(x_2))$. f is the classifier in the target domain, which can be inferred from the aligned data-label pairs.

While many approaches have been proposed, learning the mapping between different domains is still an open problem. Our work is motivated to support the development of more robust and comprehensible generative models for style transfer problems.

6.4 Method

We propose a general approach for style transfer. Let $x_i \in \mathbb{X}_i$ be a data point sampled from domain i. The goal is to learn a conversion model $p(x_j \mid x_i)$ that maps x_i to domain j ($j \neq i$) by changing its style and preserving the original content. For unpaired training data, we have marginal distributions $p(x_i)$, $p(x_j)$ instead of

the joint distribution $p(x_i, x_j)$, so it requires additional constraints to determine $p(x_j \mid x_i)$. Inspired by disentangled representation learning in [80], we assume that each example $x_i \in \mathbb{X}_i$ can be decomposed into a content code $c \in \mathcal{C}$ that encodes domain-invariant information and a style code $s_i \in \mathcal{S}_i$ that encodes domain-dependent information. $\mathcal{C}$ is shared across domains and contains the information we want to preserve. $\mathcal{S}_i$ is domain-specific and contains the information we want to change. In conversion stage, we extract the content code of x_i and recombine it with a style code randomly sampled from $\mathcal{S}_j$. A generative adversarial network (GAN) [53] is added to ensure that the converted samples are indistinguishable from the real ones.

Figure 6.1 shows the autoencoder model of style transfer with a partially shared latent space. Any pair of corresponding data points (x_i, x_j) is assumed to have a shared latent code $c \in \mathcal{C}$ and domain-specific style codes $s_i \in \mathcal{S}_i$, $s_j \in \mathcal{S}_j$. The generative model of domain i is an autoencoder that consists of a deterministic decoder $x_i = G_i(c_i, s_i)$ and two encoders $c_i = E_i^c(x_i)$, $s_i = E_i^s(x_i)$. The encoder $E_i = (E_i^c, E_i^s)$ and decoder G_i are inverse operations such that $E_i = G_i^{-1}$. To generate converted sample $x'_{j \leftarrow i}$, we just extract and recombine the content code of x_i with the style code of domain j.

$$\begin{aligned} x'_{i \leftarrow j} &= G_i(c_j, s_i) = G_i(E_j^c(x_j), s_i) \\ x'_{j \leftarrow i} &= G_j(c_i, s_j) = G_j(E_i^c(x_i), s_j) \end{aligned} \tag{6.1}$$

It should be noted that the style code s_i is not inferred from one example, but learnt from the entire target domain j, which is a major difference from [80]. This is because the style extracted from a single example is ambiguous and may not capture the general characteristics of the target domain. To restrict the mapping

$p(x_j \mid x_i)$, we use an constraint that is slightly different from the cycle consistency in [138]. It assumes that an example converted to another domain and converted back should be unchanged, i.e., $x''_{i\leftarrow j\leftarrow i} = x_i$. Instead, we apply a semi-cycle consistency in the latent space by assuming that only the latent codes remain unchanged $E_i^c(x'_{i\leftarrow j}) = c_i$ and $E_i^s(x'_{i\leftarrow j}) = s_i$. The relaxed constraint provides diversity to the generated samples.

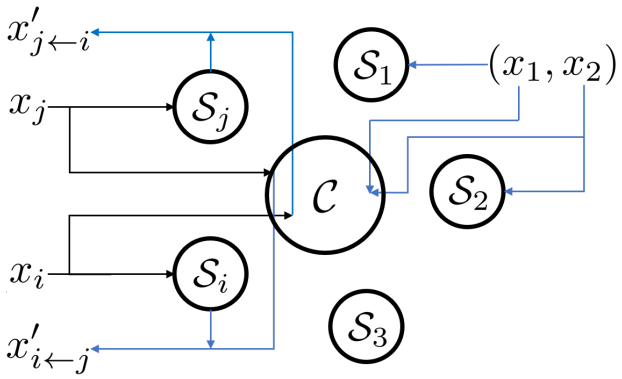

Figure 6.1: Autoencoder model with partially shared latent space. Sample x_i is encoded into a domain-specific space $\mathcal{S}_i$ and a shared content space $\mathcal{C}$. Corresponding data points (x_1, x_2) are encoded in the same content code. Converted sample $x'_{j\leftarrow i}$ is generated by recombining the original content code of x_i and the style code randomly sampled from $\mathcal{S}_j$.

We formulate the learning problem as a distributionally robust game. Each domain i has four agents E_i^c, E_i^s, G_i, D_i, in which D_i is a discriminator with two objectives. One is to distinguish between real samples and machine-generated samples, the other is to classify domain labels. On the other side, the generators G_i have two purposes: synthesize realistic samples and convert them into domain i. For example in voice conversion, the synthesized speech is evaluated on both the naturalness of its quality and the correctness of speaker's identity.

6.4.1 Two-Domain Transformation

There are $4n$ agents for n domains. For simplicity, we first investigate the conversion model between two domains. When $n = 2$, there are 8 agents: E_1^c, E_1^s, G_1, D_1 for domain $\mathbb{X}_1$ and E_2^c, E_2^s, G_2, D_2 for domain $\mathbb{X}_2$. Each agent has a different objective, and the utility function of one agent depends on the action of other agents. Collaboration and competition exist among them. So this is a distributionally robust game with cooperative agents and uncertain utility functions. Based on the analysis above, there are five modules need to learn: ① domain-invariant content encoder, ② domain-specific style encoders, ③ real/fake discriminator, ④ domain classifier D_i, ⑤ fake samples generator.

The encoders and decoders form a group to synthesize converted samples in the target domain. Another group is the discriminator and classifier. They work on the opposite side to distinguish between real/fake samples and predict the domain label. The intergroup competition and intragroup collaboration are listed in table 6.1.

When Nash equilibrium reaches, the autoencoder $(E_i^{c*}, E_i^{s*}, G_i^*)$ minimizes the reconstruction error $L_{rec}^x \to 0, L_{rec}^c \to 0, L_{rec}^s \to 0$. The GAN network (D_i^*, G_i^*) and adversarial loss $L_{GAN}^{x_i}$ converge at saddle points that minimize the distance between $p(x_i)$ and $p(x'_{j\leftarrow i})$. The classifier D_i^{cls*} correctly predicts the domain category of both real and fake samples $D_i^{cls*}(x_i) = i$, $D_i^{cls*}(x'_{i\leftarrow j}) = i$.

The collaboration means, agents in different domains can help each other as they may have common interests. For instance, a sample can be used to update the real/fake discriminator even if its class label is missing. Except for the autoencoders E_i^c, E_i^s, G_i, other agents form cross-domain coalitions. (G_1, G_2) works on data synthesis, (D_1, D_2) works on real/fake discrimination, (E_1^c, E_2^c) extracts high-level

Table 6.1: Cooperative game

Intergroup competition	Learning module	Objective
E_1^s, E_2^s	②	$\min L_{cyc}^s$
D_1, D_2	④	$\min L_{cls}^x$
Intragroup collaboration	**Learning module**	**Objective**
E_1^c, E_1^s, G_1	$Autoencoder_1$	$\min L_{rec}^{x_1}$
E_2^c, E_2^s, G_2	$Autoencoder_2$	$\min L_{rec}^{x_2}$
G_1, G_2	⑤	$\min L_{GAN}^x$
D_1, D_2	③	$\max L_{GAN}^x$
E_1^c, E_2^c	①	$\min L_{cyc}^c$

content information that we want to preserve.

We jointly train the encoders, decoders and GAN's discriminators with multiple objectives. To keep encoder and decoder as inverse operations, a reconstruction loss is applied in the direction $x_i \to (c_i, s_i) \to x_i'$, $(i, j \in 1, 2)$. Sample x_i should not be changed after encoding and decoding.

$$L_{rec}^{x_i} = \mathbb{E}_{x_i}(\|x_i - x_i'\|_1), \quad x_i' = G_i(E_i^c(x_i), E_i^s(x_i)) \tag{6.2}$$

In our model, the latent space is partially shared. Thus the cycle consistency constraint [138] is not preserved, i.e., $x''_{1\leftarrow 2\leftarrow 1} \neq x_1$. We apply a semi-cycle loss in the coding direction $c_1 \to x'_{2\leftarrow 1} \to c'_{2\leftarrow 1}$ and $s_2 \to x'_{2\leftarrow 1} \to s'_{2\leftarrow 1}$.

$$\begin{aligned} L_{cyc}^{c_1} &= \mathbb{E}_{c_1,s_2}(\|c_1 - c'_{2\leftarrow 1}\|_1), \quad c'_{2\leftarrow 1} = E_2^c(x'_{2\leftarrow 1}) \\ L_{cyc}^{s_2} &= \mathbb{E}_{c_1,s_2}(\|s_2 - s'_{2\leftarrow 1}\|_1), \quad s'_{2\leftarrow 1} = E_2^s(x'_{2\leftarrow 1}) \end{aligned} \tag{6.3}$$

Moreover, we add a GAN module to ensure the quality of generated samples. They should be indistinguishable from the real samples in the target domain $\mathbb{X}_i$. GAN loss is computed between x_j and $x'_{i\leftarrow j}$ to represent the distance between two

distributions $p(x_j)$, $p(x'_{i\leftarrow j})$.

$$L_{GAN}^{x_i} = \mathbb{E}_{c_j,s_i}[\log(1 - D_i(x'_{i\leftarrow j}))] + \mathbb{E}_{x_i}[\log D_i(x_i)] \tag{6.4}$$

The full loss is the weighted sum of L_{recon}, L_{cycle}, L_{GAN}.

$$\begin{aligned} &\min_{E_1^c,E_1^s,E_2^c,E_2^s,G_1,G_2} \max_{D_1,D_2} L(E_1^c, E_1^s, E_2^c, E_2^s, G_1, G_2, D_1, D_2) \\ &= \lambda_s(L_{cyc}^{s_1} + L_{cyc}^{s_2}) + \lambda_c(L_{cyc}^{c_1} + L_{cyc}^{c_2}) + \lambda_x(L_{rec}^{x_1} + L_{rec}^{x_2}) + \lambda_g(L_{GAN}^{x_1} + L_{GAN}^{x_2}) \end{aligned} \tag{6.5}$$

where $\lambda_s, \lambda_c, \lambda_x, \lambda_g$ control the weights of the components.

6.4.2 Multi-Domain Transformation

In multi-domain case, there are $4n$ agents in the game. To reduce complexity, we replace the domain-specific models E_i^c, G_i, D_i with a shared content encoder E^c, a shared decoder G, and a single multi-class classifier D^{cls}. Thus, only $n+3$ agents left. Figure 6.2 shows the multi-domain transformation model.

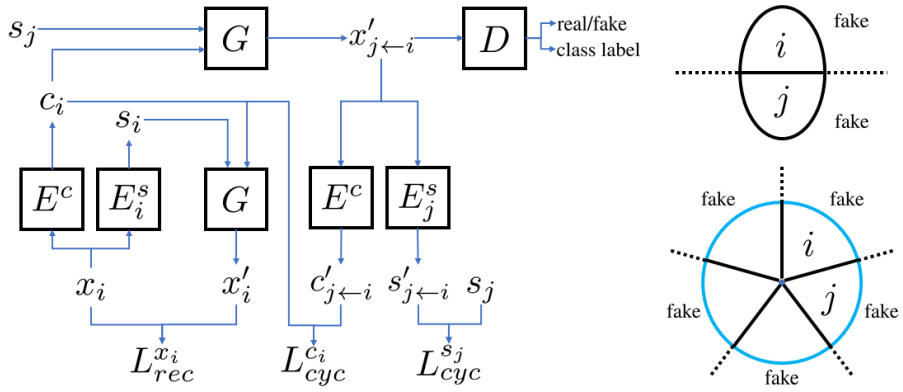

Figure 6.2: Learn multi-domain transformation. Left: conversion model with shared content encoder, decoder and classifier. Top-right: two domains. Bottom-right: multiple domains (n=5).

When $n = 2$, there are three kinds of data: real samples in $\mathbb{X}_1$, real samples in $\mathbb{X}_2$ and fake samples in $\mathbb{X}_{fake}$. Two real/fake discriminators D_1, D_2 are enough for classification. If this idea is extended to multiple domains, there will be n binary discriminators and 2^n outputs. Instead, we replace them with one binary real/fake discriminator D and one multi-class domain classifier D^{cls}. The two-step classification not only reduces complexity ($2n$ outputs) but also makes the most of the training data. An example in domain i is also useful to train the generative model for other domains, as it is the common interest of all agents $G_1, G_2, \ldots G_n$ to synthesize realistic data. Therefore, the multi-domain transformation model can be trained on different datasets with partially labeled data.

In the following section, we will apply the proposed method for emotional speech conversion.

6.5 Nonparallel Emotional Speech Conversion

Voice transformation (VT) is a technique to modify some properties of human speech while preserving its linguistic information. VT can be applied to change the speaker identity, i.e., voice conversion (VC) [155], or to transform the speaking style of a speaker, such as emotion and accent conversion [164]. In this work, we will focus on emotion voice transformation. The goal is to change emotion-related characteristics of a speech signal while preserving its linguistic content and speaker identity. Emotion conversion techniques can be applied to various tasks, such as enhancing computer generated speech, hiding negative emotions for people, helping film dubbing, and creating more expressive voice messages on social media.

Traditional VC approaches cannot be applied directly because they change

speaker identity by assuming pronunciation and intonation to be a part of the speaker-independent information. Since the speaker's emotion is mainly conveyed by prosodic aspects, some studies have focused on modelling prosodic features such as pitch, tempo, and volume [111, 165]. In [166], a rule-based emotional voice conversion system was proposed. It modifies prosody-related acoustic features of neutral speech to generate different types of emotions. A speech analysis-synthesis tool STRAIGHT [113] was used to extract fundamental frequency (F_0) and power envelope from raw audio. These features were parameterized and modified based on Fujisaki model [167] and target prediction model [168]. The converted features were then fed back into STRAIGHT to re-synthesize speech waveforms with desired emotions. However, this method requires temporal aligned parallel data that is difficult to obtain in real applications; and the accurate time alignment needs manual segmentation of the speech signal at phoneme level, which is very time consuming.

To address these issues, we propose a nonparallel training method. Instead of learning one-to-one mapping between paired emotional utterances (x_1, x_2), we switch to training a conversion model between two emotional domains $(\mathcal{X}_1, \mathcal{X}_2)$. The model presented in Section 6.4 is applied to learn the style code and content code for human speech. We evaluated our approach on IEMOCAP [169] for four emotions: angry, happy, neutral, sad; which is widely studied in emotional speech recognition literatures [170]. To our knowledge, this is the first attempt of nonparallel emotion conversion on such a large-scale dataset, though synthetic feature representations of emotional speech were proposed in [171]. We evaluate the model's conversion ability by the percentage change from source emotion to target emotion. A subjective evaluation on Amazon MTurk with hundreds of listeners was conducted. It shows

our model can effectively change emotions and retain the speaker identity.

6.5.1 Emotion-related Features

Previous emotion conversion methods directly modify parameterized prosody-related features that convey emotions. The use of Gaussian mixture models (GMM) for spectrum transformation was first proposed in [172]. [114] introduced a data-driven emotion conversion system that combines independent parameter transformation techniques including HMM-based F_0 generation, F_0 segment selection, duration conversion and GMM-based spectral conversion. However, it requires large amounts of parallel data. A recent work [166] explored four types of acoustic features: F_0 contour, spectral sequence, duration and power envelope, and investigated their impact on emotional speech synthesis by controlled feature replacement. The authors found that F_0 and spectral sequence are the dominant factors in emotion conversion, while power envelope and duration alone has little influence. They further claimed that all emotions can be synthesized by modifying the spectral sequence, but did not provide a method to do it. In this paper, we focus on learning the conversion models for F_0 and spectral sequence.

6.5.2 Nonparallel Training Approaches

Parallel data means utterances with the same linguistic content but varying in aspects to be studied. Since parallel data is hard to collect, nonparallel approaches have been developed. Some borrow ideas from image-to-image translation [139] and create GAN models [53] suitable for speech, such as VC-VAW-GAN [158], SVC-GAN [173], VC-CycleGAN [174, 175], VC-StarGAN [171]. Another trend is based on auto-regressive models like WaveNet [75]. Although it can train directly

on raw audio without feature extraction, the heavy computational load and huge amount of training data required is not affordable for most users.

6.5.3 Disentangled representation learning

Our work draws inspiration from recent studies in image style transfer. A basic idea is to find disentangled representations that can independently model image content and style. It is claimed in [127] that a Convolutional Neural Network (CNN) is an ideal representation to factorize semantic content and artistic style. They introduced a method to separate and recombine content and style of natural images by matching feature correlations in different convolutional layers. For us, the task is to find disentangled representations for speech signal that can split emotion from speaker identity and linguistic content.

6.5.4 Assumptions

The research on human emotion expression and perception has two major conclusions. First, human emotion perception is a multi-layered process. It's figured out that humans do not perceive emotion directly from acoustic features, but through an intermediate layer of semantic primitives [176]. They introduced a three-layered model and learnt the connections by a fuzzy inference system. Some researchers found that adding middle layers can improve emotion recognition accuracy [177]. Based on this finding, we suggest the use of multilayer perceptrons (MLP) to extract emotion-related information in speech signals.

Second, the emotion generation process of human speech follows the opposite direction of emotion perception. This means the encoding process of the speaker is the inverse operation of the decoding process of the listener. We assume that emo-

tional speech generation and perception share the same representation methodology, that is, the encoder and decoder are inverse operations with mirror structures.

Let $x_1 \in \mathcal{X}_1$ and $x_2 \in \mathcal{X}_2$ be utterances drawn from two different emotional categories. Our goal is to learn a mapping between two distributions $p(x_1)$ and $p(x_2)$. Since the joint distribution $p(x_1, x_2)$ is unknown for nonparallel data, the conversion models $p(x_1|x_2)$ and $p(x_2|x_1)$ cannot be directly estimated. To solve this problem, we make two assumptions:

- The speech signal can be decomposed into an emotion-invariant content code and an emotion-dependent style code;
- The encoder E and decoder G are inverse functions.

6.5.5 Model

Fig. 6.3 shows the generative model of speech with a partially shared latent space. A pair of corresponding speech (x_1, x_2) is assumed to have a shared latent code $c \in \mathcal{C}$ and emotion-related style codes $s_1 \in \mathcal{S}_1, s_2 \in \mathcal{S}_2$. For any emotional speech x_i, we have a deterministic decoder $x_i = G_i(c_i, s_i)$ and its inverse encoders $c_i = E_i^c(x_i)$, $s_i = E_i^s(x_i)$. To convert emotion, we just extract and recombine the content code of the source speech with the style code of the target emotion.

$$\begin{aligned} x'_{1\leftarrow 2} &= G_1(c_2, s_1) = G_1(E_2^c(x_2), s_1) \\ x'_{2\leftarrow 1} &= G_2(c_1, s_2) = G_2(E_1^c(x_1), s_2) \end{aligned} \tag{6.6}$$

It should be noted that the style code s_i is not inferred from one utterance, but learnt from the entire emotion domain. This is because the emotion style from a single utterance is ambiguous and may not capture the general characteristics of the

target emotion. It makes our assumption slightly different from the cycle consistent constraint [138], which assumes that an example converted to another domain and converted back should remain the same as the original, i.e., $x''_{1\leftarrow 2\leftarrow 1} = x_1$. Instead, we apply a semi-cycle consistency in the latent space by assuming that $E_1^c(x'_{1\leftarrow 2}) = c_1$ and $E_1^s(x'_{1\leftarrow 2}) = s_1$.

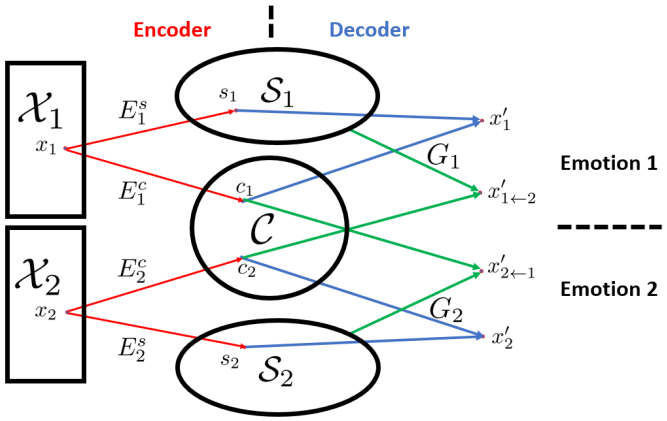

Figure 6.3: The speech autoencoder model with partially shared latent space. Speech with emotion $i$is decomposed into an emotion-specific space $\mathcal{S}_i$ and a shared content space $\mathcal{C}$. Corresponding speech (x_1, x_2) are encoded to the same content code. Emotion conversion is obtained by recombining the content code of the input utterance and the style code randomly sampled from the target emotion space.

Traditional emotional speech analysis mainly focuses on four types of acoustic features: fundamental frequency (F_0), spectral sequence, time duration and energy envelope. It was found in [166] that only F_0 and spectral sequence have significant influence, while the other two require manual segmentation and have little impact on changing emotions. Therefore we focus on learning the conversion model for F_0 and spectral sequence. Fig. 6.4 shows an overview of our nonparallel emotional speech conversion system. The features are extracted and recombined by WORLD [178] and converted separately. We modify F_0 by linear transform to match statistics of the fundamental frequencies in the target emotion domain. The conversion is

performed by log Gaussian normalization

$$f_2 = \exp((\log f_1 - \mu_1) \cdot \frac{\sigma_2}{\sigma_1} + \mu_2) \tag{6.7}$$

where μ_i, σ_i are the mean and variance obtained from the source and target emotion set. Aperiodicity (AP) is mapped directly since it does not contain emotion-related information.

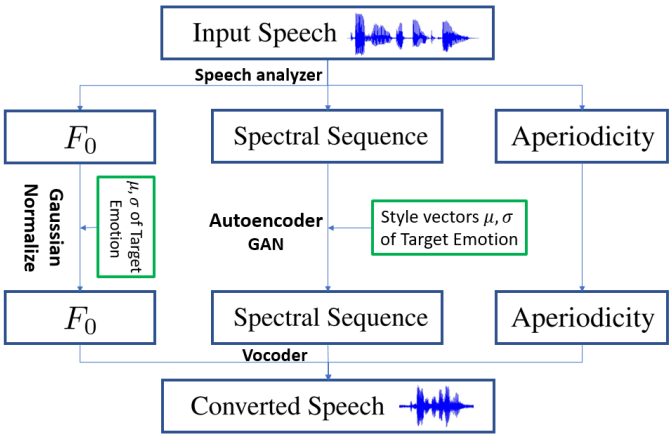

Figure 6.4: Overview of the proposed nonparallel emotion conversion system

For spectral sequence, we use low-dimensional representation in mel-cepstrum domain to reduce complexity. Kameoka [179] shows 50 MCEP coefficients are enough to synthesize full-band speech without quality degeneration. Spectra conversion is learnt by the autoencoder model in Fig. 6.3. The encoders and decoders are implemented with gated CNN [180]. In addition, a GAN module is added and trained by robust optimization [18] to produce realistic spectral frames. Our model has four subnetworks E^c, E^s, G, D, in which D is the discriminator in GAN to distinguish real samples from machine-generated samples.

6.5.6 Loss functions

We jointly train the encoders, decoders and GAN's discriminators with multiple losses displayed in Fig. 6.5. To keep encoder and decoder as inverse operations, we apply reconstruction loss in the direction $x_i \to (c_i, s_i) \to x'_i$. The spectral sequence should not change after encoding and decoding.

$$L_{recon}^{x_i} = \mathbb{E}_{x_i}(\|x_i - x'_i\|_1), \quad x'_i = G_i(E_i^c(x_i), E_i^s(x_i)) \tag{6.8}$$

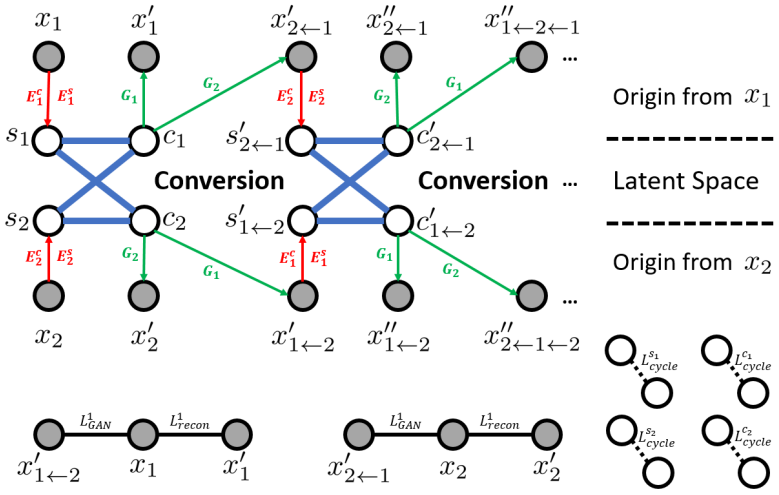

Figure 6.5: Train on multiple loss functions.

In our model, the latent space is partially shared. Thus the cycle consistency constraint [138] is not preserved, i.e., $x''_{1\leftarrow 2\leftarrow 1} \neq x_1$. We apply a semi-cycle loss in the coding direction $c_1 \to x'_{2\leftarrow 1} \to c'_{2\leftarrow 1}$ and $s_2 \to x'_{2\leftarrow 1} \to s'_{2\leftarrow 1}$.

$$\begin{aligned} L_{cycle}^{c_1} &= \mathbb{E}_{c_1,s_2}(\|c_1 - c'_{2\leftarrow 1}\|_1), \quad c'_{2\leftarrow 1} = E_2^c(x'_{2\leftarrow 1}) \\ L_{cycle}^{s_2} &= \mathbb{E}_{c_1,s_2}(\|s_2 - s'_{2\leftarrow 1}\|_1), \quad s'_{2\leftarrow 1} = E_2^s(x'_{2\leftarrow 1}) \end{aligned} \tag{6.9}$$

Moreover, we add a GAN module to improve the speech quality. The converted

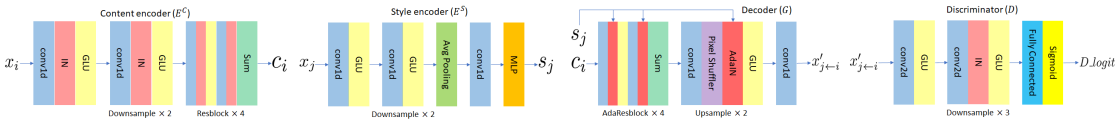

Figure 6.6: The network structure of content encoder, style encoder, decoder, and GAN discriminator

samples should be indistinguishable from the real samples in the target emotion domain. GAN loss is computed between $x'_{i\leftarrow j}$ and x_i, $(i \neq j)$.

$$L^i_{GAN} = \mathbb{E}_{c_j,s_i}[\log(1 - D_i(x'_{i\leftarrow j}))] + \mathbb{E}_{x_i}[\log D_i(x_i)] \tag{6.10}$$

The full loss is the weighted sum of L_{recon}, L_{cycle}, L_{GAN}.

$$\begin{aligned} &\min_{E^c_1,E^s_1,E^c_2,E^s_2,G_1,G_2} \max_{D_1,D_2} L(E^c_1, E^s_1, E^c_2, E^s_2, G_1, G_2, D_1, D_2) \\ &= \lambda_s(L^{s_1}_{cycle} + L^{s_2}_{cycle}) + \lambda_c(L^{c_1}_{cycle} + L^{c_2}_{cycle}) \\ &+\lambda_x(L^1_{recon} + L^2_{recon}) + \lambda_g(L^1_{GAN} + L^2_{GAN}) \end{aligned} \tag{6.11}$$

where $\lambda_s, \lambda_c, \lambda_x, \lambda_g$ control the weights of the components.

6.6 Experiments

Training emotional speech conversion models often suffers from lack of data. Parallel datasets such as Emo-DB [181] and RAVDESS [182] have limited sentence diversity and are difficult to build. Our end-to-end model is trained on raw audio signals of natural speech, and does not rely on paired data or any manual operations. Training set can be collected from daily conversations in everyday life.

6.6.1 Experiment setup

We evaluated the proposed approach on the Interactive Emotional Dyadic Motion Capture database (IEMOCAP) [169]. It is organized in five sessions and contains 12 hours of audiovisual data. Each session records natural dialogues between a pair of speakers in scripted and improvised scenarios, in which the emotions are naturally elicited. In this paper, we only consider four emotional categories: 1) angry, 2) happy, 3) neutral, 4) sad. Since the model is not designed to change the speaker identity, experiments are conducted for each speaker independently. We only use the utterances with a clear majority vote regarding the ground truth labels. There are 2754 utterances shared amongst four emotional labels: ang (747), hap (675), neu (788), sad (544). Training and testing sets are non-overlapping utterances randomly selected from the same speaker (80% for training, 20% for test). For example in session 1, there are 420 training samples and 108 testing samples for the female speaker.

Training samples with fixed length of 128 frames are randomly selected from raw audio sequences. Energy-based voice-activity detection (VAD) is used to remove silent frames. We use WORLD vocoder [126] to extract fundamental frequencies, spectral sequences (sps) and aperiodicities (aps) from raw audio waveforms sampled at 16KHz. The frame length is 5ms. After coding, we take the first 24 Mel-cepstral coefficients (MCEPs) as feature vectors. Mean and variance of the entire training set are calculated for feature normalization. Testing samples can have arbitrary temporal length, and be converted in real-time.

6.6.2 Network architecture

The network architecture is illustrated in Figure 6.6, with details listed in Table 6.6.2. The autoencoders take 24-dimentional MCEPs as input and learn disentangled representations of content and style. In the content encoder, instance normalization (IN) [183] removes the original feature mean and variance that represent emotional style information. In the style encoder, the emotional characteristics are encoded by a 3-layer MLP that outputs channel-wise mean and variance $\mu(s), \sigma(s)$. Then they are fed into the decoder to reconstruct MCEP features. The desired emotion is added through an adaptive instance normalization (AdaIN) [184] layer before activation. This mechanism is similar to the conversion model of F_0 in Eq. (6.7).

$$\text{AdaIN}(c, s) = \sigma(s)\Big(\frac{c - \mu(c)}{\sigma(c)}\Big) + \mu(s) \tag{6.12}$$

The encoders and decoders are implemented with 1D-CNNs to capture the temporal dependencies, while the GAN discriminators are implemented with 2D-CNNs to capture the spectra-temporal patterns. Higher resolution data is generated by the pixel shuffler layer in upsample blocks. All networks use gated linear units (GLU) [180] to keep track of sequential information.

Training details: We use Adam optimizer with $\beta_1 = 0.5$. The learning rate is initialized as 0.0001 for D and 0.0002 for E^c, E^s, G; it begins with linear decay applied after $150K$ iterations. Weights are chosen as $\lambda_s = \lambda_c = \lambda_g = 1$, $\lambda_x = 10$. E^c, E^s, G are trained 2 iterations for each D's iteration in the first $100K$ iterations; after that they are trained equally.

Content Encoder	
Conv1d, IN, GLU	C-128-15-1
Downsample1d × 2	C-256-5-2, C-512-5-2
Resblock1d × 4	C-512-3-1, content code
Style Encoder	
Conv1d, GLU	C-128-15-1
Downsample1d without IN × 2	C-256-5-2, C-512-5-2
Downsample1d without IN × 2	C-512-3-2
Adaptive average pooling	
Conv1d	C-16-1-1
MLP: linear × 2	flatten, dense output
Decoder	
Adaptive Resblock1d × 3	C-512-3-1
Upsample1d × 2	C-512-5-1-2, C-256-5-1-2
Conv1d	C-24-15-1, MCEPs output
Discriminator	
Conv2d	C-128-(3,3)-(1,2)
Downsample2d	C-256-(3,3)-(2,2)
Downsample2d	C-512-(3,3)-(2,2)
Downsample2d	C-1024-(6,3)-(1,2)
Dense layer	sigmoid output (real/fake)

Table 6.2: Network Architecture. C-F-K-S-X indicates convolution layer with filters F, kernel size K, strides S, and shuffle X. IN is instance normalization; all modules use GLU activation.

6.6.3 Experiment results

We evaluate the generated speech on three metrics: voice quality, speaker similarity, and the emotion conversion ability.

Subjective Evaluation We perform perception tests on Amazon Mechanical Turk [1]. Each utterance was listened by 5 random human workers, and each worker can answer at most 5 hits in a single experiment. To evaluate the voice quality and speaker similarity, the listeners were asked to give a 5-scale opinion score (5 for

[1] https://www.mturk.com

the best, 1 for the worst). The mean opinion score (MOS) is shown in Fig. 6.7. To annotate the emotion state, each listener was asked to choose a label from the source and target emotions. For example in the trial "ang2neu", utterances with label "ang" in IEMOCAP were converted to "neu", and the generated speech was labelled by the majority vote of human annotators. We compute percentage change from the source emotion to the target emotion. Higher value indicates stronger ability of emotional conversion. We choose four emotion pairs with significant differences [181]. The baseline models are a simple linear F_0 conversion system [185], and a neural network model VC-StarGAN [179]. Results are displayed in Figure 6.8. Details and some converted speech samples are provided at [2].

Results Note that not all utterances can be successfully converted, because some emotions are delivered by linguistic information, an immutable part in our setting. Our model is slightly better than VC-StarGAN in terms of emotion conversion ability (average 48% vs 44%) and speaker similarity (average 3.55 vs 3.05). One reason is that VC-StarGAN is designed for voice conversion among different speakers, while our model learns the disentangled representations that can decompose the emotional characteristic and speaker identity. Moreover, VC-StarGAN has poor voice quality in the direction of sad2ang (1.71) and sad2hap (1.81). In [179], all emotions are trained together, thus it's unfair to the sad domain since it has lower signal to noise ratio (SNR), and may amplify the noise when converted to more energetic emotions. The reason that *neu2ang* and *neu2hap* are not working as well as their inverse conversions is because neutral speech has a very broad spectrum or range compared to angry and happy, therefore the voice quality is vulnerable to the modulation noise. Increasing the resolution of the mel-cepstrums can reduce

[2] https://www.jian-gao.org/emovc

that problem, but modulation noise still remains.

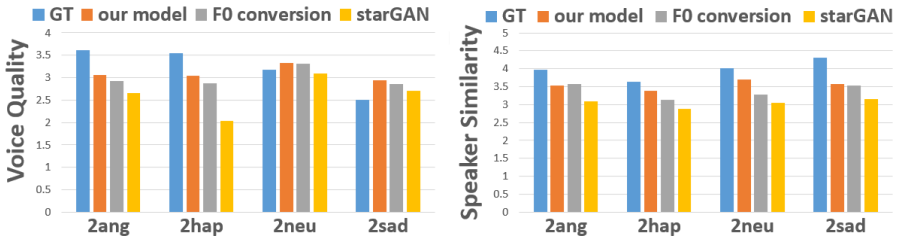

Figure 6.7: MOS for voice quality and speaker similarity. left: voice quality. right: speaker similarity, 2ang means the target emotion is Angry, and compared with originally Angry speech.

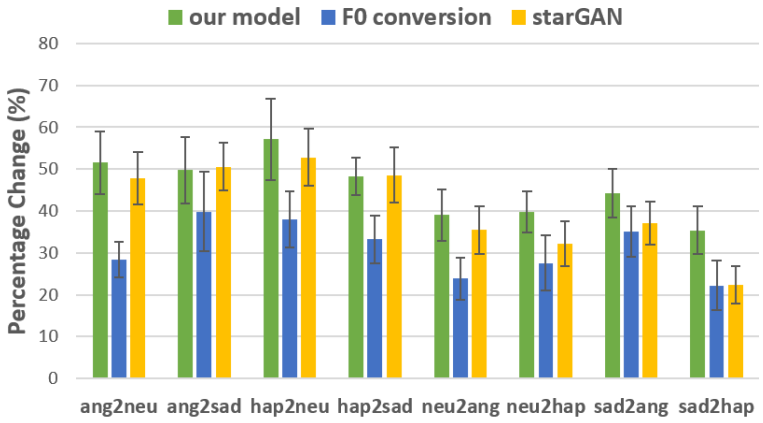

Figure 6.8: Comparison of the emotion conversion ability of our model and the baseline systems: (1) F_0 conversion, (2) VC-StarGAN [179]. ang2neu is conversion from Angry to Neutral.

6.7 Discussion

We introduced a nonparallel emotional speech conversion approach based on style transfer autoencoders. As our model does not require any paired data, transcripts or time alignment, it is easy to collect training data and apply in real-world situations. To the best of our knowledge, this is the first work on nonparallel

emotion conversion using style transfer. Future work includes phonetic duration conversion and designing a general model for unseen speakers. The method that learns the mapping of data across domains is a general framework. It can be applied to other kinds of data by replacing the encoder and decoder networks.

Chapter 7

Conclusion

In this thesis, we have investigated the problem of learning generative models, and developed several approaches from the game theoretic perspective. We consider this problem as one of the most pivotal research directions of artificial intelligence other than supervised learning. It enables the machines to better understand the rich structure of our world beyond class labels, and offers them the opportunity to think and act as humans. In particular, we have contributed in the following fields

1. We explore three widely used metrics to measure the distance between probability measures: Bregman divergence, $f-$divergence and Wasserstein distance. We compared their properties and claimed that Wasserstein distance is more theoretically sounding in stabilizing the learning process. We also introduced some methods to approximate the Wasserstein distance, and offered an example to show it supports displacement interpolation between two probability distributions. (Chapter 3)

2. We introduce a novel framework of distributionally robust games. In this game, each agent has to make decisions in an uncertain environment. To

handle the uncertainty, the agents model the state of nature by an adversarial distribution and optimize on their expected worst-case performance. The learning procedure is conducted by distributionally robust optimization. We developed Bregman learning algorithms to solve the Nash equilibrium, and proved their error bound and convergence rate. It shows exponential decay for convex objective functions, and converges fast in non-convex non-concave settings. (Chapter 4)

3. We formulate the problem of learning generative models as a distributionally robust game, and use Wasserstein distance to quantify the discrepancy between model and data distributions. The complexity of the problem is reduced by introducing an equivalent optimization problem, in which the Wasserstein distance is approximated by low-dimensional computation. We also designed practical implementations of our framework to train deep generative models for unsupervised clustering and image synthesis. (Chapter 5)

4. We developed an autoencoder model that maps data from one domain to another domain. It learns the disentangled representation of data by decomposing the high-level content information and the low-level style information. This model is a general framework that can apply to various kinds of data given customized design of decoders and encoders. Based on this idea, we designed a nonparallel emotional voice conversion system. The conversion is performed by extracting and recombining the content code of the source speech and the style code of the target emotion. It can automatically change the emotion conveyed in human speech and does not rely on paired data for training. (Chapter 6)

Unsupervised generative modeling remains rarely used in large-scale systems, since it is an ill-posed problem and works on high dimensional distributions. The good thing is, it opens the way toward real intelligence, that is, the ability to learn directly from raw data without relying on costly annotations by human. We connect this concept with game theory, where the supervisory signal is provided by the interaction between agents. The uncertainty is modeled by robust optimization in the worst-case scenarios.

This idea may have impact on other types of learning problems, for example, adversarial training. Generative models can produce additional fake samples as a source of data augmentation to protect classifiers from adversarial attacks. Improving the robustness of classification will be a major issue in supervised learning. Another topic is transfer learning. Humans learn knowledge gradually, from simple tasks to hard ones. The resource needed for learning new knowledge is much less than that for machines, because there is a "common sense" that covers knowledge in many domains. We need a universal model to learn the high-level "common sense", and algorithms to transfer knowledge between different domains.

Though the framework we proposed can handle a variety of problems, there are several limitations left for future research. First, the distributionally robust game model involves two layers of optimization problems conducted alternatively. The time complexity of estimating the dynamic payoff is as high as that of searching for the optimal strategies. Since we do not have the resource to solve the optimal transport problem in high dimensions, many approximating algorithms have been proposed. If we look back at history, the breakthrough of supervised learning comes from big model and big data, instead of a single fancy algorithm. Once we have enough computing power, the Wasserstein distance can be exactly solved. The

unsupervised approaches may perform much better than the supervised ones since they do not rely on annotations, thus the training set is unlimited and unbiased.

Second, many theoretical proofs are not valid for non-convex non-concave settings. While practical advantages have been examined in some specific tasks, we still need solid theoretical foundations so that the models can be transferred to solve multiple problems.

Third, the evaluation of generative models is still not clear. Though final results can be evaluated, there is no intrinsic performance measure during training. The dynamic payoff is just a relative indicator that depends on the performance of other players, and it does not reflect the quality of generations. Moreover, the evaluation is on the scale of distributions and cannot be computed with a single sample point. These problems make the training process unstable and difficult to monitor. A future work is to design an indicator to identify the training stability and to quantify the general performance.

Appendix A

A.1 Proof of Proposition 4.2.1

Proof. Define $\hat{g}(a_2, a_3) = \inf_{a_1 \in \mathcal{A}_1} l_3(a_1, a_2, a_3)$. Thus for all a_2, a_3 we have

$$\hat{g}(a_2, a_3) \leq l_3(a_1, a_2, a_3).$$

It follows that, for any a_1, a_3

$$\sup_{a_2 \in \mathcal{A}_2} \hat{g}(a_2, a_3) \leq \sup_{a_2 \in \mathcal{A}_2} l_3(a_1, a_2, a_3)$$

Using the definition of $\hat{g}$, one obtains

$$\sup_{a_2 \in \mathcal{A}_2} \inf_{a_1 \in \mathcal{A}_1} l_3(a_1, a_2, a_3) \leq \sup_{a_2 \in \mathcal{A}_2} l_3(a_1, a_2, a_3), \forall a_1, a_3.$$

Taking the infimum in a_1 yields

$$\sup_{a_2 \in \mathcal{A}_2} \inf_{a_1 \in \mathcal{A}_1} l_3(a_1, a_2, a_3) \leq \inf_{a_1 \in \mathcal{A}_1} \sup_{a_2 \in \mathcal{A}_2} l_3(a_1, a_2, a_3), \forall a_3. \tag{A.1}$$

Now, we use two operations for the variable a_3:

1. Taking the infimum in the inequality (A.1) in a_3 yields

$$\inf_{a_3\in\mathcal{A}_3}\sup_{a_2\in\mathcal{A}_2}\inf_{a_1\in\mathcal{A}_1} l_3(a_1,a_2,a_3) \leq \inf_{a_3\in\mathcal{A}_3}\inf_{a_1\in\mathcal{A}_1}\sup_{a_2\in\mathcal{A}_2} l_3(a_1,a_2,a_3)$$
$$= \inf_{(a_1,a_3)\in\mathcal{A}_1\times\mathcal{A}_3}\sup_{a_2\in\mathcal{A}_2} l_3(a_1,a_2,a_3)$$

   which proves the second part of the inequalities (4.9). The first part of the inequalities (4.9) follows immediately from (A.1).

2. Taking the supremum in inequality (A.1) in a_3 yields

$$\sup_{(a_2,a_3)\in\mathcal{A}_2\times\mathcal{A}_3}\inf_{a_1\in\mathcal{A}_1} l_3(a_1,a_2,a_3) \leq \sup_{a_3\in\mathcal{A}_3}\inf_{a_1\in\mathcal{A}_1}\sup_{a_2\in\mathcal{A}_2} l_3(a_1,a_2,a_3),$$

   which proves the first part of the inequalities (4.10). The second part of the inequalities (4.10) follows immediately from (A.1).

This completes the proof. □

A.2 Proof of Proposition 4.2.2

Proof. Let $\lambda^*(a)$, $\mu^*(a)$ be solution to problem (P_j^*) associated with the profile a. Then, the optimal likelihood L^* is obtained by differentiating f^* or by inverting the equation $f'(L^*) = \frac{l_j-\mu}{\lambda}$. As m is a probability measure, and using the definition of L^*, one gets

$$dm'^*(\omega) = L^* dm(\omega).$$

It follows that a_j^*, L^* solves the original problem (P_j). □

A.3 Proof of Proposition 4.3.2

Proof. Let

$$W(b(t)) = t\mathbb{E}_m[h(b^*,\omega) - h(b(t),\omega)] + d_g(b^*, b(t)),$$

where b is solution to (4.20). The function W is positive and $\frac{d}{dt}W = \mathbb{E}_m[h(b^*,\omega) - h(b(t),\omega)] - t\langle \mathbb{E}_m \nabla_b h(b,\omega), g_{bb}^{-1}\mathbb{E}_m \nabla_b h(b(t),\omega)\rangle + \frac{d}{dt}d_g(b^*, b(t))$. By concavity of $\mathbb{E}_m h(b,\omega)$ one has

$$\langle \mathbb{E}_m \nabla_b h(b,\omega), (b^* - b)\rangle \geq \mathbb{E}_m[h(b^*,\omega) - h(b,\omega)], \forall b.$$

On the other hand,

$$\begin{aligned}\frac{d}{dt}d_g(b^*, b(t)) &= -\dot{b}g_b(b) - \langle g_{bb}\dot{b}, b - b^*\rangle + g_b\dot{b} \\ &= -\langle g_{bb}\dot{b}, b - b^*\rangle \\ &= -\langle \mathbb{E}_m \nabla_b h(b,\omega), b^* - b\rangle.\end{aligned}$$

Thus,

$$\begin{aligned}\frac{d}{dt}W &\leq \langle \mathbb{E}_m \nabla_b h(b,\omega), (b^* - b)\rangle - t\langle \mathbb{E}_m \nabla_b h(b,\omega), g_{bb}^{-1}\mathbb{E}_m \nabla_b h(b,\omega)\rangle \\ &\quad - \langle \mathbb{E}_m \nabla_b h(b,\omega), b^* - b\rangle \\ &= -t\langle \mathbb{E}_m \nabla_b h(b,\omega), g_{bb}^{-1}\mathbb{E}_m \nabla_b h(b,\omega)\rangle \leq 0,\end{aligned}$$

where the last inequality is by convexity of g. It follows that $\frac{d}{dt}W(b(t)) \leq 0$ along the path of the gradient flow. This decreasing property implies $0 \leq W(b(t)) \leq$

$W(b(0)) = d_g(b^*, b(0))$. In particular,

$$0 \leq t\mathbb{E}_m[h(b^*, \omega) - h(b, \omega)] \leq W(b(0)) < +\infty.$$

Hence, the error to the value $\mathbb{E}_m h(b^*, \omega)$ is bounded by

$$0 \leq \mathbb{E}_m[h(b^*, \omega) - h(b, \omega)] \leq \frac{W(b(0))}{t}.$$

The announced result on the regret follows by integration over $[t_0, T]$ and by averaging. Note that the above regret-bound is established without assuming strong convexity of $b \mapsto -\mathbb{E}_m h(b, \omega)$, and also no Lipschitz continuity bound of the gradient is assumed. This completes the proof. □

A.4 Proof of Proposition 4.3.3

Proof. A simple computation shows that

$$\partial_x d_g(y, x) = g_x(y) - g_x(x), \quad \partial_x d_g(y, x) = -g_{xx}(x)(y - x)$$

By differentiating the functional $\hat{l}$ one gets

$$\begin{aligned}
\hat{l}_x &= e^{\alpha+\gamma}[e^{\beta} l_x - d_{g,1}(x + e^{-\alpha}, x) - d_{g,2}(x + e^{-\alpha} v, x)] \\
&= e^{\alpha+\gamma}[-g_x(y) + g_x(x) + g_{xx}(x)(y - x) + e^{\beta} l_x], \\
\hat{l}_v &= -e^{\gamma} d_{g,1}(x + e^{-\alpha} v, x) = -e^{\gamma}[g_x(x + e^{-a} v) - g_x(x)]
\end{aligned}$$

It follows that

$$\begin{aligned}\frac{d}{dt}\hat{l}_v &= -\dot{\gamma}e^{\gamma}[g_x(x+e^{-\alpha}v)-g_x(x)] \\ &\quad - e^{\gamma}[g_{xx}(x+e^{-\alpha}v)(\dot{x}-\dot{a}e^{-\alpha}\dot{x}+e^{-\alpha}\ddot{x})-g_{xx}(x)\dot{x}] \\ &= \hat{l}_x \\ &= e^{\alpha+\gamma}[-g_x(y)+g_x(x)+g_{xx}(x)(y-x)+e^{\beta}l_x]\end{aligned}$$

Then,

$$\begin{aligned}&-\dot{\gamma}[g_x(x+e^{-\alpha}\dot{x})-g_x(x)]-[g_{xx}(x+e^{-\alpha}\dot{x})(\dot{x}-\dot{\alpha}e^{-\alpha}\dot{x}+e^{-\alpha}\ddot{x})-g_{xx}(x)\dot{x}] \\ &= e^{\alpha}[-g_x(x+e^{-\alpha}\dot{x})+g_x(x)+g_{xx}(x)e^{-\alpha}\dot{x}+e^{\beta}l_x]\end{aligned}$$

Rearranging the terms in $\ddot{x}$, $\dot{x}$ one arrives at

$$\begin{aligned}&e^{\alpha}(\dot{\gamma}-e^{\alpha})g_{xx}^{-1}(x+e^{-\alpha\dot{x}})[g_x(x+e^{-\alpha}\dot{x})-g_x(x)] \\ &+e^{\alpha}g_{xx}^{-1}(x+e^{-\alpha}\dot{x})[g_{xx}(x+e^{-\alpha}\dot{x})(\dot{x}-\dot{\alpha}e^{-\alpha}\dot{x})-g_{xx}(x)\dot{x}] \\ &+e^{2\alpha}g_{xx}^{-1}(x+e^{-\alpha}\dot{x})[g_{xx}e^{-\alpha}\dot{x}+e^{\beta}l_x]+\ddot{x}=0\end{aligned}$$

By taking $\dot{\gamma}=e^{\alpha}$ it yields

$$\begin{aligned}&e^{\alpha}g_{xx}^{-1}(x+e^{-\alpha}\dot{x})[g_{xx}(x+e^{-\alpha\dot{x}})(\dot{x}-\dot{\alpha}e^{-\alpha}\dot{x})] \\ &+e^{2\alpha}g_{xx}^{-1}(x+e^{-\alpha}\dot{x})[g_{xx}e^{-\alpha}\dot{x}+e^{\beta}l_x]+\ddot{x}=0 \\ &\ddot{x}+(e^{\alpha}-\dot{a})\dot{x}+e^{2\alpha+\beta}g_{xx}^{-1}(x+e^{-\alpha}\dot{x})l_x(x)=0\end{aligned}$$

which is a second order ordinary differential system. This completes the proof. □

A.5 Proof of Proposition 4.3.4

Proof. The proof of Proposition 4.3.4 is based on a careful construction of a generalized pseudo-potential function using Pontryagin's maximum principle. It extends the framework developed in [186] to the context of strategic-form games. Then we check that the following function V is a Lyapuvov function

$$V(x(t), \dot{x}(t), t) = d_g(x^*, x(t) + e^{-\alpha(t)}\dot{x}(t)) + e^{\beta(t)}[l(x(t)) - l(x^*)]$$

where $x(t)$ is generated by the Bregman algorithm.

Since $V(x, \dot{x}, t)$ is positive and l is convex, the time derivative of V over the path $x(t), \dot{x}(t) : \frac{d}{dt}V(x(t), \dot{x}(t), t) \leq 0$ on condition that $\dot{\beta} \leq e^{\alpha}$, we have

$$e^{\beta}[l(x) - l(x^*)] \leq V(x, \dot{x}, t) \leq V(x_0, \dot{x}_0, t_0)$$

That is, the error $l(x) - l(x^*) \leq e^{-\beta}V(x_0, \dot{x}_0, t_0)$, which shows an exponential convergence to x^*. This completes the proof. □

A.6 Proof of Proposition 4.3.5

Proof. Let $V(b, \dot{b}, t, b^*) = d_g(b^*,\ b(t) + e^{-\alpha(t)\dot{b}(t)}) + e^{\beta(t)}\mathbb{E}_m[h(b^*, \omega) - h(b(t), \omega)]$. It is clear that V is positive. Moreover, $\frac{d}{dt}V(b(t), \dot{b}(t), t, b^*) \leq 0$ for $\dot{\beta} \leq e^{\alpha}$. Thus, $V(b(t), \dot{b}(t), t, b^*) \leq V(b(0), \dot{b}(0), 0, b^*) = c_0$. By integration between $[t_0, T]$ it follows

$$\frac{1}{T - t_0}\int_{t_0}^{T} \mathbb{E}_m[h(b^*, \omega) - h(b(t), \omega)]dt \leq \frac{c_0}{T - t_0}\int_{t_0}^{T} e^{-\beta(t')}dt'$$

This completes the proof. □

A.7 Proof of Proposition 4.3.6

Proof. Since $\bar{b}(t) = \frac{1}{t}\int_0^t b(t')dt' = \int_{\mathbb{R}} b(t')(\frac{1}{t}\mathbb{1}_{[0,t]}(t'))dt'$, then $\bar{b}(t) = \mathbb{E}_{\xi(t)}$ where $\xi(t)$ is the measure with density $d\xi(t)[t'] = \frac{1}{t}\mathbb{1}_{[0,t]}(dt')$. By convexity of $-\mathbb{E}_m h(b,\omega)$, we apply the Jensen's inequality:

$$\begin{aligned}\mathbb{E}_m h(\frac{1}{t}\int_0^t b(t)dt,\omega) &= \mathbb{E}_m h(\bar{b}(t),\omega) = \mathbb{E}_m h(\mathbb{E}_{\xi(t)}b,\omega)\\ &\geq \mathbb{E}_{\xi(t)}\mathbb{E}_m h(b,\omega) = \frac{1}{t}\int_0^t \mathbb{E}_m h(b(t'),\omega)dt'\end{aligned}$$

In view of (4.28), we have

$$\begin{aligned}0 &\leq \mathbb{E}_m h(b^*,\omega) - \mathbb{E}_m h(\frac{1}{t}\int_0^t b(t')dt',\omega)\\ &\leq \frac{1}{t}\int_0^t [\mathbb{E}_m h(b^*,\omega) - \mathbb{E}_m h(b(t'),\omega)]dt'\\ &\leq \frac{c_0}{t}\int_0^t e^{-\beta(t')}dt',\end{aligned}$$

That is, $$0 \leq \mathbb{E}_m h(b^*,\omega) - \mathbb{E}_m h(\bar{b}(t),\omega) \leq \frac{c_0}{t}\int_0^t e^{-\beta(t')}dt'$$

This completes the proof. □

Bibliography

[1] Hugo Touvron, Andrea Vedaldi, Matthijs Douze, and Herve Jegou. Fixing the train-test resolution discrepancy. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d Alche-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8250–8260. Curran Associates, Inc., 2019.

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[3] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. Objects as points. *CoRR*, abs/1904.07850, 2019.

[4] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *CoRR*, abs/1506.02640, 2015.

[5] Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. Multi-task deep neural networks for natural language understanding. *CoRR*, abs/1901.11504, 2019.

[6] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhut-

dinov, and Quoc V. Le. Xlnet: Generalized autoregressive pretraining for language understanding. *ArXiv*, abs/1906.08237, 2019.

[7] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Vedavyas Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy P. Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529:484–489, 2016.

[8] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

[9] Andre Esteva, Brett Kuprel, Roberto A. Novoa, Justin Ko, Susan M. Swetter, Helen M. Blau, and Sebastian Thrun. Dermatologist-level classification of skin cancer with deep neural networks. *Nature*, 542(7639):115–118, 2017.

[10] George Saon, Gakuto Kurata, Tom Sercu, Kartik Audhkhasi, Samuel Thomas, Dimitrios Dimitriadis, Xiaodong Cui, Bhuvana Ramabhadran, Michael Picheny, Lynn-Li Lim, Bergul Roomi, and Phil Hall. English conversational telephone speech recognition by humans and machines. In *Proc. Interspeech 2017*, pages 132–136, 2017.

[11] D. G. Lowe. Object recognition from local scale-invariant features. In

Proceedings of the Seventh IEEE International Conference on Computer Vision, volume 2, pages 1150–1157 vol.2, Sep. 1999.

[12] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 886–893 vol. 1, June 2005.

[13] Shawn Hershey, Sourish Chaudhuri, Daniel P. W. Ellis, Jort F. Gemmeke, Aren Jansen, Channing Moore, Manoj Plakal, Devin Platt, Rif A. Saurous, Bryan Seybold, Malcolm Slaney, Ron Weiss, and Kevin Wilson. Cnn architectures for large-scale audio classification. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2017.

[14] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *Commun. ACM*, 60(6):84–90, May 2017.

[15] S. Liu and W. Deng. Very deep convolutional neural network based image classification using small training sample size. In *2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR)*, pages 730–734, Nov 2015.

[16] Kaiqing Zhang, Zhuoran Yang, Han Liu, Tong Zhang, and Tamer Basar. Fully decentralized multi-agent reinforcement learning with networked agents. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, pages 5867–5876, 2018.

[17] Richard Bellman. Dynamic programming. *Science*, 153(3731):34–37, 1966.

[18] Jian Gao, Yida Xu, Julian Barreiro-Gomez, Massa Ndong, Michalis Smyrnakis, and Hamidou Tembine. *Distributionally Robust Optimization*. IntechOpen, Ltd., London, UK, September 2018.

[19] Hamidou Tembine. *Distributed Strategic Learning for Wireless Engineers*. CRC Press, Inc., Boca Raton, FL, USA, 2012.

[20] Michele Aghassi and Dimitris Bertsimas. Robust game theory. *Mathematical Programming*, 107(1):231–273, Jun 2006.

[21] John von Neumann, Oskar Morgenstern, and Ariel Rubinstein. *Theory of Games and Economic Behavior (60th Anniversary Commemorative Edition)*. Princeton University Press, 1944.

[22] John F. Nash. Equilibrium points in n-person games. *Proceedings of the National Academy of Sciences*, 36(1):48–49, 1950.

[23] John Nash. Non-cooperative games. *Annals of Mathematics*, 54(2):286–295, 1951.

[24] J. Gao and H. Tembine. Empathy and berge equilibria in the forwarding dilemma in relay-enabled networks. In *2017 International Conference on Wireless Networks and Mobile Communications (WINCOM)*, pages 1–8, Nov 2017.

[25] J. Gao and H. Tembine. Distributed mean-field-type filters for traffic networks. *IEEE Transactions on Intelligent Transportation Systems*, 20(2):507–521, Feb 2019.

[26] J. Gao and H. Tembine. Correlative mean-field filter for sequential and spatial data processing. In *IEEE EUROCON 2017 -17th International Conference on Smart Technologies*, pages 243–248, July 2017.

[27] J. Gao and H. Tembine. Distributed mean-field-type filter for vehicle tracking. In *2017 American Control Conference (ACC)*, pages 4454–4459, May 2017.

[28] J. Gao and H. Tembine. Distributed mean-field-type filters for big data assimilation. In *2016 IEEE 18th International Conference on High Performance Computing and Communications; IEEE 14th International Conference on Smart City; IEEE 2nd International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, pages 1446–1453, Dec 2016.

[29] Michael Maschler, Eilon Solan, and Shmuel Zamir. *Game Theory*. Cambridge University Press, 2013.

[30] Eric Rasmusen. *Games and Information An Introduction to Game Theory (4th Edition)*. Blackwell Pub., OCT 2006.

[31] J. v. Neumann. Zur theorie der gesellschaftsspiele. *Mathematische Annalen*, 100(1):295–320, Dec 1928.

[32] John C. Harsanyi. Games with incomplete information played by "bayesian" players, i-iii. part ii. bayesian equilibrium points. *Management Science*, 14(5):320–334, 1968.

[33] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the 2015 IEEE International Conference on Computer Vision*

(ICCV), ICCV '15, pages 1026–1034, Washington, DC, USA, 2015. IEEE Computer Society.

[34] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692, 2019.

[35] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[36] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *International Conference on Learning Representations*, 2019.

[37] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[38] G. E. Hinton. Deep belief networks. *Scholarpedia*, 4(5):5947, 2009. revision #91189.

[39] Ruslan Salakhutdinov and Geoffrey Hinton. Deep boltzmann machines. In David van Dyk and Max Welling, editors, *Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics*, volume 5 of

Proceedings of Machine Learning Research, pages 448–455, Hilton Clearwater Beach Resort, Clearwater Beach, Florida USA, 16–18 Apr 2009. PMLR.

[40] Rajat Kumar Sinha, Ruchi Pandey, and Rohan Pattnaik. Deep learning for computer vision tasks: A review. *CoRR*, abs/1804.03928, 2018.

[41] Fanhuai Shi, Jian Gao, and Xixia Huang. An affine invariant approach for dense wide baseline image matching. *International Journal of Distributed Sensor Networks*, 12(12):1550147716680826, 2016.

[42] Nitish Srivastava, Ruslan Salakhutdinov, and Geoffrey Hinton. Modeling documents with a deep boltzmann machine. In *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence*, UAI'13, pages 616–624, Arlington, Virginia, United States, 2013. AUAI Press.

[43] Nitish Srivastava and Ruslan Salakhutdinov. Multimodal learning with deep boltzmann machines. *J. Mach. Learn. Res.*, 15(1):2949–2980, January 2014.

[44] Jian Gao and Fanhuai Shi. A rotation and scale invariant approach for dense wide baseline matching. In De-Shuang Huang, Vitoantonio Bevilacqua, and Prashan Premaratne, editors, *Intelligent Computing Theory*, pages 345–356, Cham, 2014. Springer International Publishing.

[45] Honglak Lee, Alexis Battle, Rajat Raina, and Andrew Y. Ng. Efficient sparse coding algorithms. In *Proceedings of the 19th International Conference on Neural Information Processing Systems*, NIPS'06, pages 801–808, Cambridge, MA, USA, 2006. MIT Press.

[46] A Ng. Sparse autoencoder. In *CS294A Lecture notes*, 2011.

[47] Christophe Andrieu, Nando de Freitas, Arnaud Doucet, and Michael I. Jordan. An introduction to mcmc for machine learning. *Machine Learning*, 50(1):5–43, Jan 2003.

[48] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *J. Mach. Learn. Res.*, 11:3371–3408, December 2010.

[49] Yann LeCun, Sumit Chopra, Raia Hadsell, Fu Jie Huang, and et al. A tutorial on energy-based learning. In *PREDICTING STRUCTURED DATA*. MIT Press, 2006.

[50] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2013.

[51] Mihaela Rosca, Balaji Lakshminarayanan, and Shakir Mohamed. Distribution matching in variational inference, 2018.

[52] Lars M. Mescheder, Sebastian Nowozin, and Andreas Geiger. Adversarial variational bayes: Unifying variational autoencoders and generative adversarial networks. *CoRR*, abs/1701.04722, 2017.

[53] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014.

[54] William Fedus, Mihaela Rosca, Balaji Lakshminarayanan, Andrew M. Dai,

Shakir Mohamed, and Ian Goodfellow. Many paths to equilibrium: Gans do not need to decrease a divergence at every step, 2017.

[55] Luke Metz, Ben Poole, David Pfau, and Jascha Sohl-Dickstein. Unrolled generative adversarial networks, 2016.

[56] Xudong Mao, Qing Li, Haoran Xie, Raymond Y. K. Lau, Zhen Wang, and Stephen Paul Smolley. On the effectiveness of least squares generative adversarial networks, 2017.

[57] Ishaan Gulrajani, Faruk Ahmed, Martín Arjovsky, Vincent Dumoulin, and Aaron C. Courville. Improved training of wasserstein gans. *CoRR*, abs/1704.00028, 2017.

[58] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks, 2015.

[59] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, pages 2180–2188, USA, 2016. Curran Associates Inc.

[60] Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, Hugo Larochelle, and Ole Winther. Autoencoding beyond pixels using a learned similarity metric. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, pages 1558–1566. JMLR.org, 2016.

[61] Tim Salimans, Ian J. Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. *CoRR*, abs/1606.03498, 2016.

[62] Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. f-gan: Training generative neural samplers using variational divergence minimization. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, pages 271–279, USA, 2016. Curran Associates Inc.

[63] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pages 6629–6640, USA, 2017. Curran Associates Inc.

[64] Mikołaj Bińkowski, Dougal J. Sutherland, Michael Arbel, and Arthur Gretton. Demystifying MMD GANs. In *International Conference on Learning Representations*, 2018.

[65] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 214–223, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.

[66] Arthur Gretton, Karsten M. Borgwardt, Malte J. Rasch, Bernhard Schölkopf,

and Alexander Smola. A kernel two-sample test. *J. Mach. Learn. Res.*, 13:723–773, March 2012.

[67] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. In *International Conference on Learning Representations*, 2018.

[68] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. *CoRR*, abs/1812.04948, 2018.

[69] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi. Photo-realistic single image super-resolution using a generative adversarial network. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 105–114, July 2017.

[70] Anonymous. U-{gat}-{it}: Unsupervised generative attentional networks with adaptive layer-instance normalization for image-to-image translation. In *Submitted to International Conference on Learning Representations*, 2020. under review.

[71] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks, 2016.

[72] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Guilin Liu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. Video-to-video synthesis. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[73] Karol Gregor, Andriy Mnih, and Daan Wierstra. Deep autoregressive networks. *CoRR*, abs/1310.8499, 2013.

[74] Aäron Van Den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, pages 1747–1756. JMLR.org, 2016.

[75] Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alexander Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. In *Arxiv*, 2016.

[76] Tim Salimans, Andrej Karpathy, Xi Chen, and Diederik P. Kingma. Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications. *CoRR*, abs/1701.05517, 2017.

[77] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc., 2017.

[78] Cédric Villani. *Optimal Transport: Old and New*. Grundlehren der mathematischen Wissenschaften. Springer, 2009 edition, September 2008.

[79] Gabriel Peyré and Marco Cuturi. Computational optimal transport, 2018.

[80] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. A neural algorithm of artistic style. *CoRR*, abs/1508.06576, 2015.

[81] Dimitris Bertsimas, David B. Brown, and Constantine Caramanis. Theory and applications of robust optimization, 2010.

[82] Herbert E. Scarf. *A min-max solution of an inventory problem*. Rand Corp., Santa Monica, Calif., 1957.

[83] Maurice Sion. On general minimax theorems. *Pacific J. Math.*, 8(1):171–176, 1958.

[84] A. L. Soyster. Convex programming with set-inclusive constraints and applications to inexact linear programming. *Operations Research*, 21(5):1154–1157, 1973.

[85] J. Gao and H. Tembine. Bregman learning for generative adversarial networks. In *2018 Chinese Control And Decision Conference (CCDC)*, pages 82–89, June 2018.

[86] Léon Bottou. On-line learning in neural networks. chapter Online Learning and Stochastic Approximations, pages 9–42. Cambridge University Press, New York, NY, USA, 1998.

[87] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, 22(3):400–407, 1951.

[88] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[89] B.T. Polyak. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1 – 17, 1964.

[90] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning

applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, Nov 1998.

[91] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014.

[92] Geoffrey Hinton. Lecture 6.5 - rmsprop, coursera: Neural networks for machine learning. *Coursera, video lectures*, page 307, 2012.

[93] Matthew D. Zeiler. ADADELTA: an adaptive learning rate method. *CoRR*, abs/1212.5701, 2012.

[94] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.*, 12:2121–2159, July 2011.

[95] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.

[96] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, pages 807–814, USA, 2010. Omnipress.

[97] Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. Empirical evaluation of rectified activations in convolutional network. *CoRR*, abs/1505.00853, 2015.

[98] Z. Liu, P. Luo, X. Wang, and X. Tang. Deep learning face attributes in the wild. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 3730–3738, Dec 2015.

[99] Dario Bauso, Jian Gao, and Hamidou Tembine. Distributionally robust games: f-divergence and learning. In *Proceedings of the 11th EAI International Conference on Performance Evaluation Methodologies and Tools, VALUETOOLS 2017, Venice, Italy, December 05-07, 2017*, pages 148–155, 2017.

[100] J. Gao and H. Tembine. Distributionally robust games: Wasserstein metric. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, July 2018.

[101] Léon Bottou, Frank E. Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning, 2016.

[102] Martín Arjovsky and Léon Bottou. Towards principled methods for training generative adversarial networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

[103] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan, 2017.

[104] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *CoRR*, abs/1411.1784, 2014.

[105] Emily L. Denton, Soumith Chintala, Arthur Szlam, and Robert Fergus. Deep generative image models using a laplacian pyramid of adversarial networks. *CoRR*, abs/1506.05751, 2015.

[106] Yujia Li, Kevin Swersky, and Rich Zemel. Generative moment matching networks. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1718–1727, Lille, France, 07–09 Jul 2015. PMLR.

[107] Aude Genevay, Gabriel Peyre, and Marco Cuturi. Learning generative models with sinkhorn divergences. In Amos Storkey and Fernando Perez-Cruz, editors, *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, volume 84 of *Proceedings of Machine Learning Research*, pages 1608–1617, Playa Blanca, Lanzarote, Canary Islands, 09–11 Apr 2018. PMLR.

[108] Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 2292–2300. Curran Associates, Inc., 2013.

[109] Jean Jacques Moreau. Proximité et dualité dans un espace hilbertien. *Bulletin de la Société Mathématique de France*, 93:273–299, 1965.

[110] Yosida Kösaku. *Functional Analysis*. Springer-Verlag Berlin Heidelberg, New York City, US, 1965.

[111] Z. Wang and Y. Yu. Multi-level prosody and spectrum conversion for emotional speech synthesis. In *2014 12th International Conference on Signal Processing (ICSP)*, pages 588–593, Oct 2014.

[112] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[113] Kawahara Hideki, Masuda-Katsuse Ikuyo, and de Cheveigné Alain. Restructuring speech representations using a pitch-adaptive time–frequency smoothing and an instantaneous-frequency-based f0 extraction: Possible role of a repetitive structure in sounds1speech files available. see http://www.elsevier.nl/locate/specom1. *Speech Communication*, 27(3):187 – 207, 1999.

[114] Zeynep Inanoglu and Steve Young. Data-driven emotion conversion in spoken english. *Speech Communication*, 51(3):268 – 283, 2009.

[115] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[116] X. Yuan, P. He, Q. Zhu, and X. Li. Adversarial examples: Attacks and defenses for deep learning. *IEEE Transactions on Neural Networks and Learning Systems*, 30(9):2805–2824, Sep. 2019.

[117] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

[118] Xuanqing Liu and Cho-Jui Hsieh. Rob-gan: Generator, discriminator, and adversarial attacker. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[119] Richard Sinkhorn and Paul Knopp. Concerning nonnegative matrices and doubly stochastic matrices. *Pacific J. Math.*, 21(2):343–348, 1967.

[120] Sebastian Ruder. An overview of gradient descent optimization algorithms. *CoRR*, abs/1609.04747, 2016.

[121] Lucas Theis, Aäron van den Oord, and Matthias Bethge. A note on the evaluation of generative models, 2015.

[122] Jian Gao, Chakraborty Deep, Hamidou Tembine, and Olaitan Olaleye. Non-parallel Emotional Speech Conversion. In *Proc. Interspeech 2019*, pages 2858–2862, 2019.

[123] J. E. Kyprianidis, J. Collomosse, T. Wang, and T. Isenberg. State of the "art": A taxonomy of artistic stylization techniques for images and video. *IEEE Transactions on Visualization and Computer Graphics*, 19(5):866–885, May 2013.

[124] Michael Elad and Peyman Milanfar. Style transfer via texture synthesis. *Trans. Img. Proc.*, 26(5):2338–2351, May 2017.

[125] Jun-Yan Zhu, Philipp Krähenbühl, Eli Shechtman, and Alexei A. Efros. Generative visual manipulation on the natural image manifold. In *Proceedings of European Conference on Computer Vision (ECCV)*, 2016.

[126] Yijun Li, Ming-Yu Liu, Xueting Li, Ming-Hsuan Yang, and Jan Kautz. A closed-form solution to photorealistic image stylization. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, editors, *Computer Vision – ECCV 2018*, pages 468–483, Cham, 2018. Springer International Publishing.

[127] L. A. Gatys, A. S. Ecker, and M. Bethge. Image style transfer using convolu-

tional neural networks. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2414–2423, June 2016.

[128] Richard Zhang, Phillip Isola, and Alexei A. Efros. Colorful image colorization. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision – ECCV 2016*, pages 649–666, Cham, 2016. Springer International Publishing.

[129] Yaniv Taigman, Adam Polyak, and Lior Wolf. Unsupervised cross-domain image generation. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

[130] Takuhiro Kaneko, Kaoru Hiramatsu, and Kunio Kashino. Generative attribute controller with conditional filtered generative adversarial networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[131] Yijun Li, Chen Fang, Jimei Yang, Zhaowen Wang, Xin Lu, and Ming-Hsuan Yang. Universal style transfer via feature transforms. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pages 385–395, USA, 2017. Curran Associates Inc.

[132] T. Wang, M. Liu, J. Zhu, A. Tao, J. Kautz, and B. Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8798–8807, June 2018.

[133] Levent Karacan, Zeynep Akata, Aykut Erdem, and Erkut Erdem. Learning

to generate images of outdoor scenes from attributes and semantic layouts. *CoRR*, abs/1612.00215, 2016.

[134] Z. Zhang, Y. Song, and H. Qi. Age progression/regression by conditional adversarial autoencoder. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4352–4360, July 2017.

[135] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision – ECCV 2016*, pages 694–711, Cham, 2016. Springer International Publishing.

[136] Xiaolong Wang and Abhinav Gupta. Generative image modeling using style and structure adversarial networks. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision – ECCV 2016*, pages 318–335, Cham, 2016. Springer International Publishing.

[137] Konstantinos Bousmalis, Nathan Silberman, David Dohan, Dumitru Erhan, and Dilip Krishnan. Unsupervised pixel-level domain adaptation with generative adversarial networks. *CoRR*, abs/1612.05424, 2016.

[138] J. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2242–2251, Oct 2017.

[139] Ming-Yu Liu, Thomas Breuel, and Jan Kautz. Unsupervised image-to-image translation networks. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural*

Information Processing Systems 30, pages 700–708. Curran Associates, Inc., 2017.

[140] Xun Huang, Ming-Yu Liu, Serge Belongie, and Jan Kautz. Multimodal unsupervised image-to-image translation. In *The European Conference on Computer Vision (ECCV)*, September 2018.

[141] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. *Domain-Adversarial Training of Neural Networks*, pages 189–209. Springer International Publishing, Cham, 2017.

[142] Mingsheng Long, Yue Cao, Jianmin Wang, and Michael I. Jordan. Learning transferable features with deep adaptation networks. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pages 97–105. JMLR.org, 2015.

[143] H. Venkateswara, J. Eusebio, S. Chakraborty, and S. Panchanathan. Deep hashing network for unsupervised domain adaptation. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5385–5394, July 2017.

[144] N. Courty, R. Flamary, D. Tuia, and A. Rakotomamonjy. Optimal transport for domain adaptation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(9):1853–1865, Sep. 2017.

[145] Nicolas Courty, Rémi Flamary, Amaury Habrard, and Alain Rakotomamonjy. Joint distribution optimal transportation for domain adaptation. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and

R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 3730–3739. Curran Associates, Inc., 2017.

[146] Bharath Bhushan Damodaran, Benjamin Kellenberger, Rémi Flamary, Devis Tuia, and Nicolas Courty. Deepjdot: Deep joint distribution optimal transport for unsupervised domain adaptation. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, editors, *Computer Vision – ECCV 2018*, pages 467–483, Cham, 2018. Springer International Publishing.

[147] L. Sun, K. Li, H. Wang, S. Kang, and H. Meng. Phonetic posteriorgrams for many-to-one voice conversion without parallel data training. In *2016 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6, July 2016.

[148] M. Abe, S. Nakamura, K. Shikano, and H. Kuwabara. Voice conversion through vector quantization. In *ICASSP-88., International Conference on Acoustics, Speech, and Signal Processing*, pages 655–658 vol.1, April 1988.

[149] Y. Stylianou, O. Cappe, and E. Moulines. Continuous probabilistic transform for voice conversion. *IEEE Transactions on Speech and Audio Processing*, 6(2):131–142, March 1998.

[150] Toru Nakashika, Tetsuya Takiguchi, and Yasuo Ariki. High-order sequence modeling using speaker-dependent recurrent temporal restricted boltzmann machines for voice conversion. In *INTERSPEECH 2014, 15th Annual Conference of the International Speech Communication Association, Singapore, September 14-18, 2014*, pages 2278–2282, 2014.

[151] Z. Wu, T. Virtanen, E. S. Chng, and H. Li. Exemplar-based sparse representation with residual compensation for voice conversion. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 22(10):1506–1521, Oct 2014.

[152] D. Griffin and Jae Lim. Signal estimation from modified short-time fourier transform. In *ICASSP '83. IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 8, pages 804–807, April 1983.

[153] Daniel Erro, Iñaki Sainz, Eva Navas, and Inma Hernáez. Improved hnm-based vocoder for statistical synthesizers. In *INTERSPEECH*, 2011.

[154] Akira Tamamori, Tomoki Hayashi, Kazuhiro Kobayashi, Kazuya Takeda, and Tomoki Toda. Speaker-dependent wavenet vocoder. In *Proc. Interspeech 2017*, pages 1118–1122, 2017.

[155] Seyed Hamidreza Mohammadi and Alexander Kain. An overview of voice conversion systems. *Speech Communication*, 88:65 – 82, 2017.

[156] D. Sundermann, H. Hoge, A. Bonafonte, H. Ney, A. Black, and S. Narayanan. Text-independent voice conversion based on unit selection. In *2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings*, volume 1, pages I–I, May 2006.

[157] Feng-Long Xie, Frank K. Soong, and Haifeng Li. A kl divergence and dnn-based approach to voice conversion without parallel training sentences. In *Interspeech 2016*, pages 287–291, 2016.

[158] Chin-Cheng Hsu, Hsin-Te Hwang, Yi-Chiao Wu, Yu Tsao, and Hsin-Min

Wang. Voice conversion from unaligned corpora using variational autoencoding wasserstein generative adversarial networks. *CoRR*, abs/1704.00849, 2017.

[159] Jaime Lorenzo-Trueba, Junichi Yamagishi, Tomoki Toda, Daisuke Saito, Fernando Villavicencio, Tomi Kinnunen, and Zhenhua Ling. The voice conversion challenge 2018: Promoting development of parallel and nonparallel methods, 2018.

[160] Takuhiro Kaneko and Hirokazu Kameoka. Parallel-data-free voice conversion using cycle-consistent adversarial networks. *ArXiv*, abs/1711.11293, 2017.

[161] A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, and R. Webb. Learning from simulated and unsupervised images through adversarial training. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2242–2251, July 2017.

[162] Mingming Gong, Kun Zhang, Tongliang Liu, Dacheng Tao, Clark Glymour, and Bernhard Schölkopf. Domain adaptation with conditional transferable components. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, pages 2839–2848. JMLR.org, 2016.

[163] S. Si, D. Tao, and B. Geng. Bregman divergence-based regularization for transfer subspace learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(7):929–942, July 2010.

[164] G. Zhao, S. Sonsaat, J. Levis, E. Chukharev-Hudilainen, and R. Gutierrez-Osuna. Accent conversion using phonetic posteriorgrams. In *2018 IEEE In-*

ternational Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 5314–5318, April 2018.

[165] Miaomiao Wang, Miaomiao Wen, Keikichi Hirose, and Nobuaki Minematsu. Emotional voice conversion for mandarin using tone nucleus model – small corpus and high efficiency. 2012.

[166] Yawen Xue, Yasuhiro Hamada, and Masato Akagi. Voice conversion for emotional speech: Rule-based synthesis with degree of emotion controllable in dimensional space. *Speech Communication*, 102:54 – 67, 2018.

[167] Hiroya Fujisaki and Keikichi Hirose. Analysis of voice fundamental frequency contours for declarative sentences of japanese. *Journal of the Acoustical Society of Japan (E)*, 5(4):233–242, 1984.

[168] Y. Xue and M. Akagi. A study on applying target prediction model to parameterize power envelope of emotional speech. In *RISP workshop NCSP'16*, 2016.

[169] Carlos Busso, Murtaza Bulut, Chi-Chun Lee, Abe Kazemzadeh, Emily Mower, Samuel Kim, Jeannette N. Chang, Sungbok Lee, and Shrikanth S. Narayanan. Iemocap: interactive emotional dyadic motion capture database. *Language Resources and Evaluation*, 42(4):335, Nov 2008.

[170] M. Neumann and N. T. Vu. Improving speech emotion recognition with unsupervised representation learning on unlabeled speech. In *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7390–7394, May 2019.

[171] Saurabh Sahu, Rahul Gupta, and Carol Y. Espy-Wilson. On enhancing speech emotion recognition using generative adversarial networks. In *INTERSPEECH*, 2018.

[172] Hiromichi Kawanami, Yohei Iwami, Tomoki Toda, Hiroshi Saruwatari, and Kiyohiro Shikano. Gmm-based voice conversion applied to emotional speech synthesis. In *INTERSPEECH*, 2003.

[173] Takuhiro Kaneko, Hirokazu Kameoka, Kaoru Hiramatsu, and Kunio Kashino. Sequence-to-sequence voice conversion with similarity metric learned using generative adversarial networks. In *INTERSPEECH*, 2017.

[174] Fuming Fang, Junichi Yamagishi, Isao Echizen, and Jaime Lorenzo-Trueba. High-quality nonparallel voice conversion based on cycle-consistent adversarial network. *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5279–5283, 2018.

[175] Takuhiro Kaneko, Hirokazu Kameoka, Kou Tanaka, and Nobukatsu Hojo. Cyclegan-vc2: Improved cyclegan-based non-parallel voice conversion. *CoRR*, abs/1904.04631, 2019.

[176] Chun-Fang Huang and Masato Akagi. A three-layered model for expressive speech perception. *Speech Communication*, 50(10):810 – 828, 2008.

[177] Xingfeng Li and Masato Akagi. Multilingual speech emotion recognition system based on a three-layer model. In *INTERSPEECH*, 2016.

[178] Masanori MORISE, Fumiya YOKOMORI, and Kenji OZAWA. World: A vocoder-based high-quality speech synthesis system for real-time applications. *IEICE Transactions on Information and Systems*, E99.D(7):1877–1884, 2016.

[179] Hirokazu Kameoka, Takuhiro Kaneko, Kou Tanaka, and Nobukatsu Hojo. Stargan-vc: non-parallel many-to-many voice conversion using star generative adversarial networks. In *2018 IEEE Spoken Language Technology Workshop, SLT 2018, Athens, Greece, December 18-21, 2018*, pages 266–273, 2018.

[180] Yann N. Dauphin, Angela Fan, Michael Auli, and David Grangier. Language modeling with gated convolutional networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, pages 933–941. JMLR.org, 2017.

[181] Felix Burkhardt, Astrid Paeschke, M. Rolfes, Walter F. Sendlmeier, and Benjamin Weiss. A database of german emotional speech. In *INTERSPEECH*, pages 1517–1520. ISCA, 2005.

[182] Steven R. Livingstone and Frank A. Russo. The ryerson audio-visual database of emotional speech and song (ravdess): A dynamic, multimodal set of facial and vocal expressions in north american english. *PloS one*, 13(5):e0196391–e0196391, May 2018.

[183] Dmitry Ulyanov, Andrea Vedaldi, and Victor S. Lempitsky. Instance normalization: The missing ingredient for fast stylization. *CoRR*, abs/1607.08022, 2016.

[184] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[185] Jianhua Tao, Yongguo Kang, and Aijun Li. Prosody conversion from neutral

speech to emotional speech. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(4):1145–1154, July 2006.

[186] Andre Wibisono, Ashia C. Wilson, and Michael I. Jordan. A variational perspective on accelerated methods in optimization. *Proceedings of the National Academy of Sciences*, 113(47):E7351–E7358, 2016.