LEARNING DYNAMICS FROM DATA USING OPTIMAL TRANSPORT TECHNIQUES AND APPLICATIONS

A Dissertation
Presented to
The Academic Faculty

By

Shaojun Ma

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
Computational Science and Engineering
College of Sciences
Department of Mathematics

Georgia Institute of Technology

August 2022

LEARNING DYNAMICS FROM DATA USING OPTIMAL TRANSPORT TECHNIQUES AND APPLICATIONS

Thesis committee:

Dr. Luca Dieci
School of Mathematics
Georgia Institute of Technology

Dr. Xiaojing Ye
Department of Mathematics and Statistics
Georgia State University

Dr. Molei Tao
School of Mathematics
Georgia Institute of Technology

Dr. Hongyuan Zha
School of Data Science
The Chinese University of Hong Kong, Shenzhen

Dr. Wuchen Li
School of Mathematics
University of South Carolina

Dr. Haomin Zhou, Advisor
School of Mathematics
Georgia Institute of Technology

Date approved: June 1st, 2022

Enjoy the fruits of discipline, strength comes with practice.

For my parents

ACKNOWLEDGMENTS

It has been six years since I became a student in Georgia Tech. My major was Mechanical Engineering because I wanted to build iron man suits, later on I realized that the core parts in iron man suits are computer science and mathematics, so I decided to explore the world that I never saw. It is such a great pleasure for me to know everybody in Tech, without their help I will never make this achievement.

Firstly I want to express my full, deepest gratitude to my advisor, Professor Haomin Zhou, who keeps leading me to build understandings of mathematics, providing meaningful research directions and supporting me in writing papers. I trust Professor Zhou for his expertise in mathematics, incisive and novel understanding of the research, as well as his nice personalities. During my PhD study in Math Department, I enjoy discussing different kinds of problems with him, including not only cutting edges of the research but also life experience, advice and encouragement. I will never forget the moment we had lunch together in China and every formula we wrote on the board in his office, it is such a great luck for me to have him as my PhD and life advisor.

I also want to convey my super gratitude to my co-advisor, Professor Hongyuan Zha, who also gives a lot of cares to my research, life and work. Professor Zha provides me a lot of interesting research directions, he always know the most advanced and popular research topics in both computational science and mathematics fields, his advice inspires to touch the most newest and interesting research directions. Professor Zha also helped me a lot while I was looking for internships and jobs, his wisdom, calm and positiveness always light my research and life.

I am extremely grateful to Professors Xiaojing Ye, Luca Dieci and Molei Tao for sparing time to serve as my committee members. I appreciate their valuable and insightful questions and suggestions.

I would like to thank Professor Wuchen Li and Professor Yongxin Chen, my collabo-

rators Shu Liu and Jiaojiao Fan for the guides and discussions. They offer me professional advice and thoughts which tremendously push forward the progress of the projects. From our communications I am able to see new possibilities, together we come up with new ideas and solve problems. I really enjoy working with them.

Meanwhile, I want to offer my thanks to my graduate fellows and friends. Allow me to specially thank Haoming Jiang, Ruilin Li, Haodong Sun, Xinshi Chen, Haoran Sun, Yuqin Yang, Yingjie Qian and Hao Wu for years of friendship and many interesting discussions in both research and life.

Last but not least, I convey my deep thanks to my parents, without the constant supporting and caring of whom, I would not have the opportunity to have gone so far.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Optimal Transport has been studied widely in recent years, the concept of Wasserstein distance brings a lot of applications in computational mathematics, machine learning, engineering, even finance areas. Meanwhile, people are gradually realizing that as the amount of data as well as the needs of utilizing data increase vastly, data-driven models have great potentials in real-world applications. In this thesis, we apply the theories of OT and design data-driven algorithms to form and compute various OT problems. We also build a framework to learn inverse OT problem. Furthermore, we develop OT and deep learning based models to solve stochastic differential equations, optimal control, mean field games related problems, all in data-driven settings.

In Chapter 2, we provide necessary mathematical concepts and results that form the basis of this thesis. It contains brief surveys of optimal transport, stochastic differential equations, Fokker-Planck equations, deep learning, optimal controls and mean field games.

Chapter 3 to Chapter 5 present several scalable algorithms to handle optimal transport problems within different settings. Specifically, Chapter 3 shows a new saddle scheme and learning strategy for computing the Wasserstein geodesic, as well as the Wasserstein distance and OT map between two probability distributions in high dimensions. We parametrize the map and Lagrange multipliers as neural networks. We demonstrate the performance of our algorithms through a series of experiments with both synthetic and realistic data.

Chapter 4 presents a scalable algorithm for computing the Monge map between two probability distributions since computing the Monge maps remains challenging, in spite of the rapid developments of the numerical methods for optimal transport problems. Similarly, we formulate the problem as a mini-max problem and solve it via deep learning. The performance of our algorithms is demonstrated through a series of experiments with both synthetic and realistic data.

In Chapter 5 we study OT problem in an inverse view, which we also call Inverse OT (IOT) problem. IOT also refers to the problem of learning the cost function for OT from observed transport plan or its samples. We derive an unconstrained convex optimization formulation of the inverse OT problem. We provide a comprehensive characterization of the properties of inverse OT, including uniqueness of solutions. We also develop two numerical algorithms, one is a fast matrix scaling method based on the Sinkhorn-Knopp algorithm for discrete OT, and the other one is a learning based algorithm that parameterizes the cost function as a deep neural network for continuous OT. Our numerical results demonstrate promising efficiency and accuracy advantages of the proposed algorithms over existing state-of-the-art methods.

In Chapter 6 we propose a novel method using the weak form of Fokker Planck Equation (FPE) — a partial differential equation — to describe the density evolution of data in a sampled form, which is then combined with Wasserstein generative adversarial network (WGAN) in the training process. In such a sample-based framework we are able to learn the nonlinear dynamics from aggregate data without explicitly solving FPE. We demonstrate our approach in the context of a series of synthetic and real-world data sets.

Chapter 7 introduces the application of OT and neural networks in optimal density control. Particularly, we parametrize the control strategy via neural networks, and provide an algorithm to learn the strategy that can drive samples following one distribution to new locations following target distribution. We demonstrate our method in both synthetic and realistic experiments, where we also consider perturbation fields.

Finally Chapter 8 presents applications of mean field game in generative modeling and finance area. With more details, we build a GAN framework upon mean field game to generate desired distribution starting with white noise. We also provide a various hidden states model to predict the daily trading volume of stocks in the stock trading markets.

CHAPTER 1
INTRODUCTION

Optimal transport (OT) problem was formalized by the French mathematician Gaspard Monge in 1781 [1]. In the 1940s The Major advances of OT problem were made by the Soviet mathematician and economist Leonid Kantorovich [2]. Hence the problem is also known as the Monge–Kantorovich problem. OT problem is a constrained optimization problem to find the optimal transport plan to move mass from initial to target locations with minimum cost. Gradually the nice geometric structures of OT were discovered by mathematicians [3, 4], and OT has became a popular and classical research topic in optimization, probability theory and finance. In recent years, OT theories have been applied in the field of partial differential equations [4], fluid dynamics [5] and differential geometry [6]. Also, due to the ability to measure the discrepancy between different distributions, OT vastly draws attentions of machine learning, robotics and control communities, we have witnessed a great success of OT application in deep learning [7, 8], robotics [9, 10], control [11, 12] and economy[13].

1.1 Computational Methods for OT and IOT Problems

We present three major works related to the computation of OT problem from Chapter 2 to Chapter 5, including two novel algorithms of handling high dimensional cases with different cost functions, and one inverse view of the OT problem (IOT), where we compute the cost function when OT plan is already given.

- **Learning Wasserstein Geodesics**

  We propose a new formulation and learning strategy for computing the Wasserstein geodesic between two probability distributions within a high dimensional setting. By

applying the method of Lagrange multipliers to the dynamic formulation of the optimal transport (OT) problem, we derive a minimax problem whose saddle point is the Wasserstein geodesic. We then parametrize the functions by deep neural networks and design a sample based bidirectional learning algorithm for training. The trained networks enable sampling from the Wasserstein geodesic. As by-products, the algorithm also computes the Wasserstein distance and OT map between the marginal distributions. We demonstrate the performance of our algorithms through a series of experiments with both synthetic and real-world data.

- **Learning Monge Map**

  Monge map refers to the optimal transport map between two probability distributions and it provides a principled approach to transform one distribution to another. In spite of the rapid developments of the numerical methods for optimal transport problems, computing the Monge maps remains challenging, especially for high dimensional problems. In this topic, we present a scalable algorithm for computing the Monge map between two probability distributions. Our algorithm is based on a weak form of the optimal transport problem, thus it only requires samples from the marginals instead of their analytic expressions, and can accommodate optimal transport between two distributions with different dimensions. Compared with other existing methods for estimating Monge maps using samples, which usually adopt quadratic costs, our algorithm is suitable for general cost functions. The performance of our algorithms is demonstrated through a series of experiments with both synthetic and realistic data.

- **Learning Cost Function in Inverse Optimal Transport**

  Inverse optimal transport (OT) refers to the problem of learning the cost function for OT from observed transport plan or its samples. In this topic, we derive an unconstrained convex optimization formulation of the inverse OT problem, which can be further augmented by any customizable regularization. We provide a comprehensive characteriza-

tion of the properties of inverse OT, including uniqueness of solutions. We also develop two numerical algorithms, one is a fast matrix scaling method based on the Sinkhorn-Knopp algorithm for discrete OT, and the other one is a learning based algorithm that parameterizes the cost function as a deep neural network for continuous OT. The novel framework proposed in the work avoids repeatedly solving a forward OT in each iteration which has been a thorny computational bottleneck for the bi-level optimization in existing inverse OT approaches. Numerical results demonstrate promising efficiency and accuracy advantages of the proposed algorithms over existing state-of-the-art methods.

1.2 Applications of OT in Data Driven Problems

Next in Chapter 6 and Chapter 7, we apply OT onto two important problems, namely, solving drift term of stochastic differential equation (SDE) and optimal density control. Unlike traditional methods, we solve problems in sample level within different dimensional settings, by leveraging neural networks. Lastly in Chapter 8, we develop a new generative adversarial network (GAN) base on mean field game (MFG). We also try to solve a popular trading problem in the final Chapter.

- **Learning Stochastic Differential Equation and Hamiltonian System**

  Learning nonlinear dynamics from aggregate data is a challenging problem because the full trajectory of each individual is not available, namely, the individual observed at one time may not be observed at the next time point, or the identity of individual is unavailable. This is in sharp contrast to learning dynamics with full trajectory data, on which the majority of existing methods are based. We propose a novel method using the weak form of Fokker Planck Equation (FPE) — a partial differential equation — to describe the density evolution of data in a sampled form, which is then combined with Wasserstein generative adversarial network (WGAN) in the training process. In such a sample-based framework we are able to learn the nonlinear dynamics from aggregate data without explicitly solving FPE. Furthermore, we extend our model to Hamiltonian system within an

aggregate setting, more specifically, we parametrize Hamiltonian H by neural network and learn it by minimizing accumulate Wasserstein distance between generated data and observed data. We demonstrate our approach in the context of a series of synthetic and real-world data sets.

- **Learning Optimal Density Control**

  Optimal control problems have been studied in a lot of areas in recent years. Typical individual-level control framework aims to control individuals precisely, the number of agents to be controlled is small. As for the density control problems, the densities are usually known in advance, which can be hardly constrained in realities, especially when surrounding environment is complex. In this paper, we provide an idea to parametrize the control as neural network to realize density control. Our work can be concluded as: 1) Review the popular optimal control frameworks. 2) Design a data-driven model to learn the optimal density control strategy $u(x)$ to drive one distribution to another distribution in sample level. 3) We apply our model on realistic application, for example path planning of a group of underwater vehicles in the ocean flow field.

- **MFG GAN and Predicting Intraday Trading Volume**

  We propose a novel mean field games (MFGs) based GAN(generative adversarial network) framework. To be specific, we utilize the Hopf formula in density space to rewrite MFGs as a primal-dual problem so that we are able to train the model via neural networks and data samples. Our model is flexible due to the freedom of choosing various functionals within the Hopf formula. Moreover, our formulation mathematically avoids Lipschitz-1 constraint. The correctness and efficiency of our method are validated through several experiments.

  Predicting intraday trading volume plays an important role in trading alpha research. Existing methods such as rolling means (RM) and a two-states based Kalman Filtering method have been presented in this topic. We extend two states into various states in

Kalman Filter framework to improve the accuracy of prediction. Specifically, for different stocks we apply cross validation and determine best states number by minimizing mean squared error of the trading volume prediction. We demonstrate the effectiveness of our method through a series of comparison experiments and numerical analysis.

CHAPTER 2
PRELIMINARY IN MATHEMATICS

In this chapter, we introduce the basic mathematics needed in this thesis, including optimal transport (OT), Fokker Planck equation (FPE), Hamiltonian system, optimal control and Mean field game (MFG). These are highly related topics and form the bases of our algorithms.

2.1 Optimal Transport

The optimal transport theory is a popular and famous branch of modern mathematics to study how to transport from one probability distribution to another one with the optimal cost. The original optimal transport problem was proposed by Monge [1] as following:

$$C_(\mu,\nu) = \inf_{T} \left\{ \int_X c(x, T(x)) d\mu(x) \right\}, \tag{2.1}$$

over the set of all measurable maps $T : X \to Y$ such that $T_\sharp \mu = \nu$. Here $T_\sharp \mu$ is the push forward measure defined as $T_\sharp \mu(A) = \mu(T^{-1}(A))$ for all measurable set $A \subset X$. We define cost function $c(x, y)$ as the effort or energy for moving one unit mass from x to y with $x \in \mu$ and $y \in nu$. μ, ν are two probability measures supported on continuous states. A relaxation version of OT problem is given as

$$C(\mu,\nu) = \inf_{\pi \in \Pi(\mu,\nu)} \left\{ \iint_{X\times Y} c(x, y)\, d\pi(x, y) \right\}, \tag{2.2}$$

where $\Pi(\mu, \nu)$ is the joint probability measure of μ and ν, namely, $\Pi(\mu, \nu) := \{\pi : \pi(\cdot, Y) = \mu, \pi(X, \cdot) = \nu\}$. Compared with Monge problem, here we do not find the optimal map T but the optimal transport plan π, which is also known as Kantorovich prob-

lem [2]. Moreover, (2.2) also admits a duality form, named Kantorovich duality [6], is presented as

$$C(\mu,\nu) = \sup_{\phi(y)-\psi(x)\leq c(x,y)} \left\{ \int_Y \phi(y)d\nu(y) - \int_X \psi(x)d\mu(x) \right\}. \tag{2.3}$$

Either (2.1), (2.2) or (2.3) can be regarded as the static view of OT problem, the reason is that none of them involve any time dynamics. If we study OT problem with an extra time dimension, we come to the dynamical formulation of optimal transport problem as

$$C(\rho_a,\rho_b) = \min_{\rho,v} \left\{ \int_0^1 \int_{\mathbb{R}^d} L(v(x,t))\ \rho(x,t)\ dxdt \right\}, \tag{2.4}$$

$$\text{subject to: } \frac{\partial \rho(x,t)}{\partial t} + \nabla \cdot (\rho(x,t)v(x,t)) = 0, \tag{2.5}$$

$$\rho(\cdot,0) = \rho_a,\ \rho(\cdot,1) = \rho_b, \tag{2.6}$$

where L can be treated as the transporting cost or as the kinetic energy of the transport motion, for example $L(v(x,t)) = \frac{1}{2}||v(x,t)||^2$, which is known as dynamical OT problem, as well as the Benamou-Brenier formulation [5]. And from above (2.5) is continuity constraint and (2.6) is boundary constraint, which describe on the time axis, the density evolution of ρ along the flow field $v(\cdot,t)$. Hence the goal of dynamical OT problem is to find optimal density evolution and flow field to continuously transfer density ρ_a to ρ_b with minimum energy cost. We also call the optimal $\{\rho(\cdot,t)\}$ as the **geodesic** (w.r.t. the cost L) joining boundary densities ρ_a and ρ_b. We will introduce more details in Chapter 3 and Chapter 4.

2.2 Fokker Planck Equation

For a normal stochastic differential equation

$$d\boldsymbol{X}_t = g(\boldsymbol{X}_t,t)dt + \sigma(\boldsymbol{X}_t,t)d\boldsymbol{W}_t, \tag{2.7}$$

where in the space $\mathbb{R}^D$, $d\boldsymbol{X}_t$ represents an infinitesimal change of $\{\boldsymbol{X}_t\}$ along with time increment dt, $g(\cdot,t) = (g^1(\cdot,t),...,g^D(\cdot,t))^T$ is the drift term of SDE, σ is the diffusion coefficient, $\{\boldsymbol{W}_t\}$ is the standard Brownian motion. We also call (2.7) as Itô process. Suppose $\phi(x,t)$ is a solution of Itô process, by Itô formula we have

$$d\phi(X) = \left(\frac{\partial\phi}{\partial\boldsymbol{x}}dX + \frac{1}{2}\sigma^2\frac{\partial^2\phi}{\partial\boldsymbol{x}^2}dX^2\right) = \left(g\frac{\partial\phi}{\partial\boldsymbol{x}} + \frac{1}{2}\sigma^2\frac{\partial^2\phi}{\partial\boldsymbol{x}^2}\right)dt + \sigma\frac{\partial\phi}{\partial\boldsymbol{x}}dW.$$

Then we take expectation on both sides gives

$$d\mathbb{E}\left[\phi\right] = \mathbb{E}\left[g\frac{\partial\phi}{\partial x} + \frac{1}{2}\sigma^2\frac{\partial^2\phi}{\partial x^2}\right]dt,$$

$$\int\phi\frac{\partial\rho}{\partial t}dx = \int g\frac{\partial\phi}{\partial x}\rho(\boldsymbol{x},t)dx + \int\frac{1}{2}\sigma^2\frac{\partial^2\phi}{\partial x^2}\rho(\boldsymbol{x},t)dx.$$

Use integration by parts on right side and consider $p(\infty,t) = 0$, we have

$$\int\phi\frac{\partial\rho}{\partial t}dx = -\int\phi\frac{\partial}{\partial x}\left[g(\boldsymbol{x})\rho(\boldsymbol{x},t)\right]dx + \int\frac{1}{2}\sigma^2\phi\frac{\partial^2\rho(\boldsymbol{x},t)}{\partial x^2}dx,$$

which leads to Fokker Planck Equation (FPE) in $\mathbb{R}^1$

$$\frac{\partial\rho}{\partial t} = -\frac{\partial}{\partial x}\left[g(\boldsymbol{x})\rho(\boldsymbol{x},t)\right] + \frac{1}{2}\sigma^2\frac{\partial^2\rho(\boldsymbol{x},t)}{\partial x^2}, \tag{2.8}$$

and from (2.8) we derive Fokker Planck Equation in $\mathbb{R}^D$ as

$$\frac{\partial\rho(\boldsymbol{x},t)}{\partial t} = \sum_{i=1}^{D} -\frac{\partial}{\partial x_i}\left[g^i(\boldsymbol{x},t)\rho(\boldsymbol{x},t)\right] + \frac{1}{2}\sigma^2\sum_{i=1}^{D}\frac{\partial^2}{\partial {x_i}^2}\rho(\boldsymbol{x},t). \tag{2.9}$$

Specially if $g(x)$ is the gradient of a potential function $V(x)$, namely, $g(x) = -\nabla V(x)$ and $\sigma = \sqrt{2D}$, it is not possible to calculate the time dependent solution of this equation for an arbitrary potential. But there exists an stationary solution for the corresponding

Fokker-Planck equation

$$\rho^* = \frac{1}{z} e^{-\frac{V(x)}{D}}, \qquad Z = \int_{\mathbb{R}^d} e^{-\frac{V(x)}{D}} dx. \tag{2.10}$$

where ρ^* is the unique invariant distribution called the Gibbs distribution, Z is the normalization factor, also called the partition function. In this thesis we are not looking for the equilibrium of the FPE, we only care the distribution evolution along time axis and study the dynamics that drives one distribution to another distribution.

2.3 Hamiltonian System

2.3.1 Definition

A system of differential equations is called a Hamiltonian system if there exists a real valued function $H(x, y)$ such that

$$\frac{dx}{dt} = \frac{\partial H}{\partial y}, \qquad \frac{dy}{dt} = -\frac{\partial H}{\partial x}, \tag{2.11}$$

for all x and y, and the function H is called the Hamiltonian function for the system. $H(x, y)$ is also considered to be a conserved quantity for the system. The Hamiltonian system often has a physical meaning for the system of ODEs that is modelling a particular real-world situation, since it represents a quantity that is being conserved over time.

To get a closer look of Hamiltonian system, let's start with an undamped harmonic oscillator

$$m\ddot{x} + kx = 0,$$

which can be rewritten as

$$\dot{x} = v, \qquad \dot{v} = -\frac{k}{m}x \tag{2.12}$$

Suppose that $(x(t), v((t))$ is a solution curve in the x-v plane. We calculate the slope of the solution curve dv/dx using the fact from calculus that the derivative of an inverse function is

$$\frac{d}{dx}f^{-1}(x) = \frac{1}{f'(x)},$$

thus we have

$$\frac{dv}{dx} = -\frac{kx}{mv},$$

which leads to the solution

$$\frac{1}{2}mv^2 + \frac{1}{2}kx^2 = H, \tag{2.13}$$

where H is a constant. The first term in (2.13) is the kinetic energy function of the harmonic oscillator and the second term is the potential energy function. We call the total energy of the harmonic oscillator as H. The above equation also tells us that in a Hamiltonian system, the energy (the sum of the potential energy and the kinetic energy) is always conserved. If we compare (2.13) with (2.11), we will find out $dx/dt = \partial H/\partial v = mv$ and $dv/dt = \partial H/\partial x = kx$, which is the same system with (2.12) by change of variable.

Generally speaking, in a formal Hamiltonian system we would like to replace the notations v with p and x with q. Other systems such as mathematical pendulum which can be described as $H(p, q) = \frac{1}{2}p^2 - \cos q$, thus the dynamics is $dp/dt = -\sin q$ and $dq/dt = p$. For Kepler problem, when computing the motion of two bodies (planet and sun) which

attract each other, we choose one of the bodies (sun) as the centre of our coordinate system, the motion will then stay in a plane and we can use two-dimensional coordinates $q = (q1, q2)$ for the position, and $p = (p1, p2)$ for the velocity of the second body, the system is depicted by $H(p_1, p_2, q_1, q_2) = \frac{1}{2}(p_1^2 + p_2^2) - 1/(\sqrt{q_1^2 + q_2^2})$. Another typical example is Hénon–Heiles problem, with $H(p, q) = \frac{1}{2}(p_1^2 + p_2^2) + \frac{1}{2}(q_1^2 + q_2^2) + q_1^2 q_2 - \frac{1}{3}q_2^3$, which is a Hamiltonian system that can have chaotic solutions, but it is out of our scope and we won't discuss more in this thesis.

2.3.2 Computing Methods

If we want to know the behaviour of Hamiltonian systems we need to compute the evolution of p and q along time axis. We briefly introduce some typical numerical methods [14] in this section.

Explicit Euler Method. The simplest of all numerical methods for the Hamiltonian system is the method formulated by Euler that reads

$$\begin{aligned} p(t+h) &= p(t) - h\frac{\partial H(p(t), q(t))}{\partial q}, \\ q(t+h) &= q(t) + h\frac{\partial H(p(t), q(t))}{\partial p}, \end{aligned} \tag{2.14}$$

the method uses a constant step size h to compute one step and one after until to the step we want, starting from a given initial value $p(0)$ and $q(0)$.

Implicit Euler Method. As an alternative to the explicit Euler scheme, it is also called the backward Euler scheme

$$\begin{aligned} p(t+h) &= p(t) - h\frac{\partial H(p(t+h), q(t))}{\partial q}, \\ q(t+h) &= q(t) + h\frac{\partial H(p(t+h), q(t))}{\partial p}, \end{aligned} \tag{2.15}$$

we see $p(t + h)$ shows in both side in the first equation, one may use Newton method to

compute approximate value.

Symplectic Euler Method. Explicit Euler Method may cause instability issues while Implicit Euler Method involves extra steps caused by Newton method. Symplectic Euler Method provides a more convenience way by

$$\begin{aligned} p(t+h) &= p(t) - h\frac{\partial H(p(t), q(t))}{\partial q}, \\ q(t+h) &= q(t) + h\frac{\partial H(p(t+h), q(t))}{\partial p}, \end{aligned} \tag{2.16}$$

Störmer–Verlet Method. When the Hamiltonian is conservative and separable, we can use this simple and widely-used symplectic integrator

$$\begin{aligned} p(t+\frac{1}{2}h) &= p(t) - \frac{1}{2}h\frac{\partial H(p(t), q(t))}{\partial q}, \\ q(t+h) &= q(t) + h\frac{\partial H(p(t+\frac{1}{2}h), q(t))}{\partial p}, \\ p(t+h) &= p(t+\frac{1}{2}h) - \frac{1}{2}h\frac{\partial H(p(t), q(t+h))}{\partial q}, \end{aligned} \tag{2.17}$$

which is also known as the leapfrog method. It is also computationally efficient as Euler's method yet considerably more accurate.

2.4 Optimal Control

We consider a stochastic control problem where the state X_s of the system is governed by the SDE with values in $\mathbb{R}^d$

$$X_s = x + \int_t^s b(\tau, X_\tau, \alpha_\tau)d\tau + \int_t^s \sigma(\tau, X_\tau, \alpha_\tau)dW_\tau, \tag{2.18}$$

where $X \in \mathbb{R}^d$, α is the control for some fixed set $\mathcal{A}$, W is a given N-dimensional Brownian motion, $b : [0, T] \times \mathbb{R}^d \times A \to \mathbb{R}^d$ and $\sigma : [0, T] \times \mathbb{R}^d \times A \to \mathbb{R}^{d\times N}$. Each controller

controls the process X through the control α to reach control goals. If define

$$J(t,x,\alpha) = \mathbb{E}\left[\int_t^T r(\tau, X_\tau, \alpha_s)d\tau + g(X_T)\right], \tag{2.19}$$

where T is the finite time horizon, and we have running cost $r : [0,T] \times \mathbb{R}^d \times A \to \mathbb{R}$ and terminal cost $g : \mathbb{R}^d \to \mathbb{R}$. We aim to find the optimal control α to minimize J by introducing the value function the map $u : [0,T] \times \mathbb{R}^d \to \mathbb{R}$, which is

$$u(t,x) = \inf_{\alpha \in \mathcal{A}} J(t,x,\alpha). \tag{2.20}$$

2.5 Mean Field Game

Base on the optimal control problem, when we deal with infinitely many agents, suppose each agent controls his or her own dynamics by

$$X_s = x + \int_t^s b(\tau, X_\tau, \alpha_\tau, \rho(\tau))d\tau + \int_t^s \sigma(\tau, X_\tau, \alpha_\tau, \rho(\tau))dW_\tau, \tag{2.21}$$

where $X \in \mathbb{R}^d$, α is the control, W is a given N-dimensional Brownian motion. The difference with (2.18) is that here all parameters are dependent to the distribution $\rho(\tau)$. The coefficients $b : [0,T] \times \mathbb{R}^d \times A \times \mathcal{P}(\mathbb{R}^d) \to \mathbb{R}^d$ and $\sigma : [0,T] \times \mathbb{R}^d \times A \times \mathcal{P}(\mathbb{R}^d) \to \mathbb{R}^{d \times N}$. The cost function is defined as

$$J(t,x,\alpha) = \mathbb{E}\left[\int_t^T r(s, X_s, \alpha_s)ds + g(X_T, \rho(T))\right], \tag{2.22}$$

where T is the finite time horizon, and we have running cost $r : [0,T] \times \mathbb{R}^d \times A \times \mathcal{P}(\mathbb{R}^d) \to \mathbb{R}$ and terminal cost $g : \mathbb{R}^d \times \mathcal{P}(\mathbb{R}^d) \to \mathbb{R}$. The value function is defined as

$$u(t,x) = \inf_{\alpha \in \mathcal{A}} J(t,x,\alpha), \tag{2.23}$$

where $u(t,x)$ solves the Hamilton-Jacobi equation

$$\begin{cases} -\partial_t u(t,x) + H(t,x,Du(x,t),D^2u(x,t),\rho(t)) = 0 \\ u(T,x) = g(x,\rho(T)), \end{cases} \tag{2.24}$$

in which the Hamiltonian $H : [0,T] \times \mathbb{R}^d \times \mathbb{R}^d \times \mathbb{R}^{d\times d} \times \mathcal{P}(\mathbb{R}^d) \to \mathbb{R}$ is defined as

$$\begin{aligned} H(t,x,Du,D^2u,\rho) = \sup_{\alpha\in\mathcal{A}}\Big\{ & -r(t,x,\alpha,\rho) - Du(t,x)b((t,x,\alpha,\rho) \\ & - \frac{1}{2}\text{Tr}(\sigma\sigma^*(t,x,\alpha,\rho)D^2u(t,x))\Big\}. \end{aligned} \tag{2.25}$$

If we denote $\alpha^*(t,x)$ as one solution of above equation, we have

$$\begin{aligned} H(t,x,Du,D^2u,\rho) = & -r(t,x,\alpha^*,\rho) - Du(t,x)b((t,x,\alpha^*,\rho) \\ & - \frac{1}{2}\text{Tr}(\sigma\sigma^*(t,x,\alpha^*,\rho)D^2u(t,x)). \end{aligned} \tag{2.26}$$

Similarly for the individual level of control at optimum we have

$$dX_\tau^* = b(\tau,X_\tau^*,\alpha^*(\tau,X_\tau^*),\rho(\tau))d\tau + \sigma(\tau,X_\tau^*,\alpha^*(\tau,X_\tau^*),\rho(\tau))dW_\tau. \tag{2.27}$$

Thus we apply Itô formula to derive density evolution $\rho(\tau)$ by

$$\begin{aligned} \mathbb{E}\Big[\phi(T,X_T^*)\Big] =& \mathbb{E}\Big[\phi(0,X_0^*)\Big] + \int_0^T \mathbb{E}\Big[\partial_\tau\phi(\tau,X_\tau^*) + b(\tau,X_\tau^*,\alpha^*(\tau,X_\tau^*),\rho(\tau))D\phi(\tau,X_\tau^*) \\ & + \frac{1}{2}\text{Tr}(\sigma\sigma^*(\tau,X_\tau^*,\alpha^*(\tau,X_\tau^*),\rho(\tau))D^2\phi(\tau,X_\tau^*)\Big] \\ =& \int_X \phi(0,x)\rho(0)dx + \int_0^T\int_X \Big[\partial_\tau\phi(\tau,x) + b(\tau,x,\alpha^*(\tau,x),\rho(\tau))D\phi(\tau,x) \\ & + \frac{1}{2}\text{Tr}(\sigma\sigma^*(\tau,x,\alpha^*(\tau,x),\rho(\tau))D^2\phi(\tau,x)\Big]\rho(\tau,x)d\tau. \end{aligned} \tag{2.28}$$

After integration by parts we will have for $\rho(t)$

$$\begin{cases} \partial_t \rho - \frac{1}{2}\sum_{i,j} D^2_{ij}(\rho(t,x)\sigma_{ij}\sigma^*_{ij}(t,x,\alpha(t,x)^*,\rho(t))) + \nabla\cdot(\rho(t,x)b(t,x,\alpha(t,x)^*,\rho(t))) = 0, \\ \rho(0,\cdot) = \rho_0. \end{cases} \tag{2.29}$$

Collecting (2.24) and (2.29) together gives MFG system

$$\begin{cases} -\partial_t u(t,x) + H(t,x,Du(x,t),D^2u(x,t),\rho(t)) = 0 \\ \partial_t \rho - \frac{1}{2}\sum_{i,j} D^2_{ij}(\rho(t,x)\sigma_{ij}\sigma^*_{ij}(t,x,\alpha(t,x)^*,\rho(t))) + \nabla\cdot(\rho(t,x)b(t,x,\alpha(t,x)^*,\rho(t))) = 0, \\ \rho(0,\cdot) = \rho_0, \qquad u(T,x) = g(x,\rho(T)), \end{cases} \tag{2.30}$$

We can simplify this system by setting $\sigma = \sqrt{2}I_d$, then we have

$$H(t,x,Du(t,x),\rho) = \sup_{\alpha\in\mathcal{A}}\Big\{-r(t,x,\alpha,\rho) - Du(t,x)b(t,x,\alpha,\rho)\Big\}, \tag{2.31}$$

by the Envelope Theorem we notice that

$$D_pH(t,x,Du(t,x),\rho(t)) = -b(t,x,\alpha(t,x),\rho(t)) \tag{2.32}$$

Finally the MFG system comes to

$$\begin{cases} -\partial_t u(t,x) - \Delta u(t,x) + H(t,x,Du(x,t),\rho(t)) = 0 \\ \partial_t \rho - \Delta\rho(t,x) - \nabla\cdot(\rho(t,x)D_pH(t,x,Du(t,x),\rho(t))) = 0, \\ \rho(0,\cdot) = \rho_0, \qquad u(T,x) = g(x,\rho(T)). \end{cases} \tag{2.33}$$

CHAPTER 3

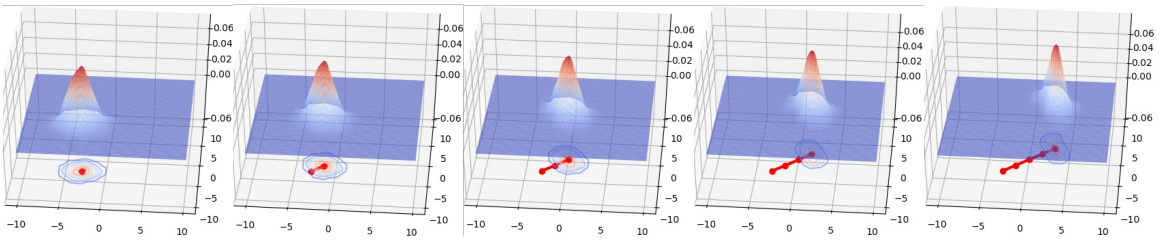
LEARNING HIGH DIMENSIONAL WASSERSTEIN GEODESICS

3.1 Introduction

As a key concept of optimal transport (OT) [15], Wasserstein distance has been widely used to evaluate the distance between two distributions. Suppose we are given any two probability distributions ρ_a and ρ_b on $\mathbb{R}^d$, recap from (2.2) that the Wasserstein distance between ρ_a and ρ_b is defined as

$$\inf_{\pi} \left\{ \int_{\mathbb{R}^d \times \mathbb{R}^d} c(x,y) d\pi(x,y) \middle| \pi \in \Pi(\rho_a, \rho_b) \right\}. \tag{3.1}$$

The solution of OT refers to an optimal π^* to attain the Wasserstein distance between two distributions as well as an optimal map T^* such that $T^*(x)$ and y have the same distribution when $x \sim \rho_a$.



(a) t=0 (b) t=0.25 (c) t=0.5 (d) t=0.75 (e) t=1

Figure 3.1: Wasserstein geodesic between two Gaussians

When applying OT-related computations to applications, the Kantorovich duality form has been widely studied and applied. For example, many regularized OT problems have been investigated, including entropic regularized OT [16, 8], Laplacian regularization [17], Group-Lasso regularized OT [18], Tsallis regularized OT [19] and OT with L^2 regularization [20].

Based on the Kantorovich form (2.4), specially if we choose cost function as $c(x, y) = \frac{1}{2}|x - y|^2$, then it leads to the well-known 2-Wasserstein distance. The OT problem can be rewritten in a fluid dynamics perspective (3.2) [21]: W_2 defined below is the square of 2-Wasserstein distance,

$$\begin{aligned} W_2(\mu, \nu) =& \inf \left\{ \int_{\mathbb{R}^d} \int_0^1 \rho(x,t)|v(x,t)|^2 dx dt \right\}, \\ &\text{subject to: } \partial_t \rho + \nabla \cdot (\rho v) = 0, \quad \rho(\cdot, 0) = \rho_a, \ \ \rho(\cdot, 1) = \rho_b. \end{aligned} \tag{3.2}$$

Solving (3.2) leads to the following partial differential equation (PDE) system:

$$\begin{aligned} &\partial_t \rho + \nabla \cdot (\rho \nabla \Phi) = 0, \quad \frac{\partial \Phi}{\partial t} + \frac{1}{2}|\nabla \Phi|^2 = 0, \\ &\text{subject to: } \rho(\cdot, 0) = \rho_a, \quad \rho(\cdot, 1) = \rho_b. \end{aligned} \tag{3.3}$$

The solution of Problem (3.2) or (3.3) gives the definition of Wasserstein *geodesic*: if there exists an optimal pushforward map Ξ^* between ρ_a and ρ_b, then the constant speed geodesic is defined as a continuous curve: $\rho_t = (\text{Id} + t\Xi^*)_\sharp \rho_a$[1], which depicts the trajectory of $x_0 \in \rho_a$ moving towards $x_1 \in \rho_b$. Here Id stands for the identity map.

Wasserstein geodesic between ρ_a and ρ_b, shown in Figure 3.1 as an example, depicts trajectory of the distribution mean (indicated by a red line), and provides ample information for the Wasserstein distance and optimal pushforward map. More importantly, Wasserstein geodesic offers a natural sampling mechanism without any artificial regularization to generate samples not only for the target distribution ρ_b, but also for all distributions along the geodesic, due to the reason that the geodesic is automatically kinetic-energy-minimizing. Our model is different from several recent OT based models for computing the optimal pushforward map, such as Jacobian and Kinetic regularized OT [22] and L^2 regularized OT [23].

[1] Please check Definition 3.2.1.

Researchers from robotics and optimal control communities also apply Wasserstein geodesic in different scenarios. For examples, [9] apply Brenier-Benamou OT to swarm control and updates the velocity of each agent. [10] study the locations of robots by minimizing the Wasserstein distance between original and target distributions. Diverse OT applications in control theories are investigated in the work of[11, 12]. However, most of current research that combines OT with robotics or control is still limited to low dimensions. Due to this reason, we suggest having a method to compute the Wasserstein geodesic, especially within high dimensional settings, will be beneficial for developing novel high dimensional algorithms and applications in robotics and control, such as path planning for multi-agent systems in high dimensional dynamics.

Also, in the domain of applied mathematics, efficient computing method of Wasserstein geodesic is important and challenging. It is well-known that solving Problem (3.2) directly or (3.3) by the traditional numerical PDE methods, such as finite difference or finite element method which requires spatial discretization, which raise up the *curse of dimensionality* issue, which is a hard problem that the computational cost grows exponentially as the dimension increases.

Finally, in this chapter our contents are organized as follows: first we formulate the OT problem as a saddle point problem without adding any regularizers. Next we reduce the search space for the saddle point problem by leveraging KKT conditions. We then parametrize the optimal maps as well as the Lagrange multipliers via deep neural networks. Our model is a sample-based algorithm that is capable of handling both low and high dimensional Wasserstein geodesic. It is also worth mentioning that there is no Lipschitz or convexity constraint on the neural networks in our computation. The constraints are thorny issues since in most of current research they can only be approximately enforced when dealing with Wasserstein related problems. Furthermore, our formulation is based on a Lagrangian framework which can be readily generalized to various alternative cost functions, including the L^p-Wasserstein distance. To the best of our knowledge, there is no method

to compute high dimensional Wasserstein distance, optimal map as well as Wasserstein geodesic all at once for a general cost function, in the sense of scalable computation. Our model is tested through a series experiments on both synthetic and realistic data sets. To summarize, our contributions are:

- We propose a novel saddle scheme so that both low and high dimensional Wasserstein geodesic, optimal map as well as Wasserstein distance between two given distributions can be computed in one single framework.
- Our scheme is able to handle general convex cost functions, including the general L^p-Wasserstein distance. Moreover, the scheme does not require convexity or Lipschitz constraint.
- The effectiveness of our method is demonstrated through extensive numerical experiments, including both synthetic and realistic data sets.

Related work: Traditional approaches [21, 24, 25, 26] for OT problems are designed to handle low dimensional settings but become infeasible in high dimensional cases since those methods are based on spatial discretizations. In machine learning community, researchers utilize OT to drive one distribution to another, and the Sinkhorn algorithm has been widely adopted [27, 28, 29, 30] due to its convenient implementation even in high dimensional settings. However, the algorithm does not scale well to a large number of samples or can't readily handle continuous probability measures [31].

To overcome the challenges of Sinkhorn algorithm, researchers bring neural networks to study OT problems. [8] study the regularized OT by applying neural networks such that large scale and high dimensional OT can be computed. [7] propose Wasserstein-1 GAN, which is a generative model to minimize Wasserstein distance. Though we have witnessed a great success of Wasserstein-1 GAN and its related work in high dimensional and large scale applications [32, 33, 34, 35, 36, 37], the non-trivial Lipschitz-1 constraint of the discriminator is hard to be theoretically satisfied.

In order to avoid Lipschitz-1 constraint in Wasserstein-1 GAN, by using input convex neural networks (ICNN) [38] to approximate the potential function, [39] propose Wasserstein-2 GAN to rewrite OT as a minimization problem, [40] and [41] extend the semi-dual formulation of OT to new minimax problems.

[42] propose a machine learning framework for solving general Mean-Field Games in high dimensional spaces. Such method can be applied to solving Optimal Transport problems by adding penalty term to enforce the terminal constraint. Many other OT based approaches have been proposed to drive one distribution to another, including continuous normalizing flows (CNF) [43, 44] and the approaches developed by [45, 46].

Various OT models bring numerous applications in domain adaptation [8], generative modeling [7], partial differential equations [47], stochastic control [48], robotics [9, 10], as well as color transfer [39], which is also one of the experiments in this chapter.

Although Wasserstein distance and optimal map have been widely studied within diverse frameworks, most of them focus on the cases that the cost function is either L^1 or L^2 based, and Wasserstein geodesic is seldom computed, especially in high dimensional settings. In this chapter we compute all in our novel designed framework. We also note that a similar strategy formulated by [49] derives a saddle point optimization scheme for solving the mean field game equations. We should point out that our problem setting and sampling method are distinct from them.

3.2 Background of the Wasserstein Geodesic

We consider two probability distributions ρ_a, ρ_b defined on $\mathbb{R}^d$. For ease of discussion, we assume the density functions of these two distributions exist, then we focus on computing an interpolation curve $\{\rho_t\}_{t=0}^1$ between ρ_a and ρ_b. To be more precise, we aim to find the length-minimizing curve joining ρ_a and ρ_b, which is also known as the *Wasserstein geodesic*. To this end, we start from the classical Optimal Transport (OT) problem (3.1),

and assume the cost function $c(x, y)$ is the optimal value of the following control problem,

$$c(x,y) = L(y-x) = \min_{\{v(\cdot,t)\}} \left\{ \int_0^1 L(v_t)\, dt \right\},$$
$$\text{subject to: } \dot{x}_t = v(x_t, t),\ x_0 = x,\ x_1 = y.$$

Here we assume the Lagrangian $L(\cdot) \in C^1(\mathbb{R}^d)$ satisfies $L(-u) = L(u)$ for arbitrary $u \in \mathbb{R}^d$ and is strictly convex and superlinear.[2] Then $\nabla L : \mathbb{R}^d \to \mathbb{R}^d$ is an invertible map, we denote ∇L^{-1} the inverse of ∇L in the sequel. We define the Hamiltonian $H(\cdot)$ associated with Lagrangian $L(\cdot)$ as

$$H(p) = \max_{v \in \mathbb{R}^d} \{v \cdot p - L(v)\} = \nabla L^{-1}(p) \cdot p - L(\nabla L^{-1}(p)), \tag{3.4}$$

This is useful in our future discussion.

Remark 1. *One important instance of L is $L(v) = \frac{|v|^p}{p}$, $p > 1$, which leads to p-Wasserstein distance. When $p = 2$, it recovers the classical 2-Wasserstein distance.*

Since (3.1) doesn't involve time, we denote it as **Static OT** problem and we denote its optimal value as $W_{\text{Static}}(\rho_a, \rho_b)$. Before introducing the dual problem of (3.1), we introduce the following definition.

Definition 3.2.1. *Given measurable map $T : \mathbb{R}^d \to \mathbb{R}^d$ and λ as a probability distribution on $\mathbb{R}^d$. We denote the pushforward of λ by T as $T_\sharp \lambda$, which is defined as*

$$T_\sharp \lambda(E) = \lambda(T^{-1}(E)) \quad \textit{for all measurable set } E \subset \mathbb{R}^d.$$

Problem (3.1) has the Kantorovich dual form [6]

$$\max_{\phi(y) - \psi(x) \le c(x,y)} \left\{ \int \phi(y) \rho_b(y)\, dy - \int \psi(x) \rho_a(x)\, dx \right\}. \tag{3.5}$$

[2] $L(\cdot)$ is super linear if $\lim_{u \to \infty} \frac{L(u)}{|u|} = \infty$.

Here and in the sequel, we will denote $\int$ as $\int_{\mathbb{R}^d}$ for conciseness. One can show that the optimal value of (3.5) equals $W_{\text{Static}}(\rho_a, \rho_b)$. Let us denote the optimizer to (3.5) as (ϕ^*, ψ^*). Then $\nabla\psi^*$ (as well as $\nabla\phi^*$) provides optimal transport maps from ρ_a to ρ_b (as well as from ρ_b to ρ_a) in the sense that

$$(\text{Id} + \nabla L^{-1}(\nabla\psi^*))_\sharp \rho_a = \rho_b,\ (\text{Id} - \nabla L^{-1}(\nabla\phi^*))_\sharp \rho_b = \rho_a. \tag{3.6}$$

We then consider the dynamic version of (3.1)

$$W_{\text{Dym}}(\rho_a, \rho_b) = \min_{\rho, v} \left\{ \int_0^1 \int L(v(x,t))\ \rho(x,t)\ dxdt \right\}, \tag{3.7}$$

$$\text{subject to: } \frac{\partial \rho}{\partial t} + \nabla \cdot (\rho v) = 0, \rho(\cdot, 0) = \rho_a, \rho(\cdot, 1) = \rho_b. \tag{3.8}$$

This problem also has an equivalent particle control version

$$\min_v \left\{ \int_0^1 \mathbb{E}[L(v(\boldsymbol{X}_t, t))]\ dt \right\}, \quad \text{subject to } \frac{d}{dt}\boldsymbol{X}_t = v(\boldsymbol{X}_t, t),\ \boldsymbol{X}_0 \sim \rho_a,\ \boldsymbol{X}_1 \sim \rho_b. \tag{3.9}$$

Such particle control version is more helpful for designing sample based formulation than its PDE counterpart, (3.7) as we will stated in later Section 3.3.2. Since we introduce the dynamics of $\{\rho_t\}$ and $\{\boldsymbol{X}_t\}$ into the new definition and reformulate the original problem (3.1) as an optimal control problem (3.7) and (3.9), we thus denote (3.7), (3.9) as **Dynamical OT** problem. The optimal solution of Dynamical OT is given by the following coupled PDE system (Chapter 13 of [6]).

$$\begin{aligned} &\frac{\partial \rho(x,t)}{\partial t} + \nabla \cdot (\rho(x,t) \nabla L^{-1}(\nabla\Phi(x,t))) = 0,\ \ \frac{\partial \Phi(x,t)}{\partial t} + H(\nabla\Phi(x,t)) = 0, \\ &\text{subject to: } \rho(\cdot, 0) = \rho_a,\ \rho(\cdot, 1) = \rho_b. \end{aligned} \tag{3.10}$$

We denote the solution to (3.10) as (ρ^*, Φ^*). Then the optimal vector field is

$$v^*(x,t) = \nabla L^{-1}(\nabla \Phi^*(x,t)). \tag{3.11}$$

From a geometric perspective, Problem (3.7) can be treated as a computing scheme for the geodesic on the probability manifold equipped with Wasserstein distance $W_{\text{Dym}}(\cdot,\cdot)$. Following this point of view, we also treat (3.7) as the problem for evaluating the **Wasserstein geodesic** joining ρ_a and ρ_b. Meanwhile, the PDE system (3.10) serves as the geodesic equation for the Wasserstein geodesic.

Remark 2. *Static OT problems and Dynamical OT problems are closely related.*

Recall (ϕ^*, ψ^*) as optimal solution to (3.5), and Φ^* as solution of (3.10), then

$$\Phi^*(x,0) = \psi^*(x) + C_0, \quad \Phi^*(x,1) = \phi^*(x) + C_1. \tag{3.12}$$

Here C_0, C_1 are constants. Furthermore, from (3.6) we have

$$(\text{Id} + \nabla L^{-1}(\nabla \Phi^*(\cdot,0)))_\sharp \rho_a = \rho_b, \quad (\text{Id} - \nabla L^{-1}(\nabla \Phi^*(\cdot,1)))_\sharp \rho_b = \rho_a. \tag{3.13}$$

In addition, Static and Dynamical OT produce the same distance, i.e. $W_{\text{Static}}(\rho_a, \rho_b) = W_{\text{Dym}}(\rho_a, \rho_b)$. Recall the original (static) optimal transport problem and consider its Kantorovich duality [6]

$$\max_{\phi(y)-\psi(x)\le c(x,y)} \left\{ \int \phi(y)\rho_b(y)\, dy - \int \psi(x)\rho_a(x)\, dx \right\} \tag{3.14}$$

Then the optimal value of (3.5) equals $W_{\text{Static OT}}(\rho_a, \rho_b)$. Let us denote the optimizer to (3.5) as (ϕ^*, ψ^*). Then $\nabla\psi^*$ (as well as $\nabla\phi^*$) will provide optimal transporting maps from

ρ_a to ρ_b (as well as from ρ_b to ρ_a), i.e. we have

$$(\mathrm{Id} + \nabla\psi^*)_\sharp \rho_a = \rho_b, \quad (\mathrm{Id} - \nabla\phi^*)_\sharp \rho_b = \rho_a. \tag{3.15}$$

For the dynamical OT problem, its optimal solution is given by

$$\begin{aligned} &\frac{\partial \rho(x,t)}{\partial t} + \nabla \cdot (\rho(x,t)\, \nabla L(\cdot)^{-1}(\nabla\Phi(x,t))) = 0 \\ &\frac{\partial \Phi(x,t)}{\partial t} + H(\nabla\Phi(x,t)) = 0 \\ &\text{subject to } \rho(\cdot,0) = \rho_a, \quad \rho(\cdot,1) = \rho_b. \end{aligned} \tag{3.16}$$

And the optimal vector field

$$v^*(x,t) = \nabla L(\cdot)^{-1}(\nabla\Phi(x,t)) \tag{3.17}$$

Both (3.10) and (3.11) can be deduced from the KKT conditions of the dynamical OT.

The connections between Static OT and Dynamical OT is tied by the following relations:

$$\Phi(x,1) = \phi^*(x) + C_1 \quad \Phi(x,0) = \psi^*(x) + C_0 \tag{3.18}$$

where C_0, C_1 are constants.

Furthermore, by (3.6), we also have

$$(\mathrm{Id} + \nabla\Phi(\cdot,0))_\sharp \rho_a = \rho_b, \quad (\mathrm{Id} - \nabla\Phi(\cdot,1))_\sharp \rho_b = \rho_a. \tag{3.19}$$

Such equations (3.6) and (3.13) are called Monge-Ampere equations. Assume $(\mathrm{Id} + \nabla u)_\sharp \rho_a = \rho_b$ admits unique solution $u(x)$, then $\nabla u = \nabla\phi^* = \nabla\Phi_0$ is the optimal transport field to the OT problem from ρ_a to ρ_b.

We also have $W_{\text{Static OT}}(\rho_a, \rho_b) = W_{\text{Dym OT}}(\rho_a, \rho_b)$. Detailed discussion on the equivalent relations between Static OT and Dynamical OT problems has also been elaborated in

several references such as section 5.1 of [15] and Chapter 13 of [6].

3.3 Proposed Methods

3.3.1 Primal-Dual based saddle point scheme

In this section, we present an approach of solving **Dynamical OT** problem (3.7) by applying Lagrange Multiplier method. We introduce Lagrange Multiplier $\Phi(x,t)$ for the PDE constraint $\frac{\partial \rho(x,t)}{\partial t} + \nabla \cdot (\rho(x,t)v(x,t)) = 0$ and $\Psi(x)$ for one of the boundary constraints $\rho(\cdot,1) = \rho_b$. (The constraint $\rho(\cdot,0) = \rho_a$ can be naturally treated as the initial condition of ρ_t.) Then we consider the functional

$$\begin{aligned}\mathfrak{L}(\rho, v, \Phi, \Psi) &= \int_0^1 \int L(v)\ \rho + \left(\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho v)\right) \Phi(x,t)\ dxdt + \int \Psi(x)(\rho(x,1) - \rho_b(x))\ dx \\ &= \int_0^1 \int \left(L(v) - \frac{\partial \Phi}{\partial t} - \nabla \Phi \cdot v\right)\ \rho(x,t)\ dxdt + \int \Phi(x,1)\rho(x,1)\ dx \\ &\quad - \int \Phi(x,0)\rho_a(x)\ dx + \int \Psi(x)(\rho(x,1) - \rho_b(x))\ dx.\end{aligned} \tag{3.20}$$

For the second equality, we apply integration by parts on $[0,1]$ and use the initial condition $\rho(\cdot,0) = \rho_a$. Solving the constrained optimization problem (3.7) is equivalent to investigating the following saddle point optimization problem

$$\min_{\rho,v} \max_{\Phi,\Psi}\ \mathfrak{L}(\rho, v, \Phi, \Psi). \tag{3.21}$$

Problem (3.21) contains (ρ, v, Φ, Ψ) as variables. We eliminate some of the variables by leveraging the Karush–Kuhn–Tucker (KKT) conditions [50]

$$\frac{\partial \mathfrak{L}}{\partial \Phi(x,t)} = 0, \frac{\partial \mathfrak{L}}{\partial \Psi(x)} = 0, \frac{\partial \mathfrak{L}}{\partial \rho(x,t)} = 0, \frac{\partial \mathfrak{L}}{\partial v(x,t)} = 0. \tag{3.22}$$

The first two conditions lead to the constraints (3.8). The third condition in (3.22) yields

$$-\frac{\partial \Phi}{\partial t} - (\nabla \Phi(x,t) \cdot v(x,t) - L(v(x,t))) = 0, \tag{3.23}$$

$$\Phi(x,1) + \Psi(x) = 0. \tag{3.24}$$

The fourth condition in (3.22) yields $\nabla L(v(x,t)) - \nabla \Phi(x,t) = 0$, which can be rewritten as

$$v(x,t) = \nabla L^{-1}(\nabla \Phi(x,t)). \tag{3.25}$$

The KKT conditions (3.24) and (3.25) reveal explicit relations among v, Φ and Ψ, which can be incorporated in (3.21) by plugging $\Psi(x) = -\Phi(x,1)$ and $v(x,t) = \nabla L^{-1}(\nabla \Phi(x,t))$ back into (3.20). Recall definition of H in (3.4), the first term of (3.20) then becomes $-\left(\frac{\partial \Phi(x,t)}{\partial t} + H(\nabla \Phi(x,t))\right)$. Our new optimization problem can be formulated as

$$\min_{\rho} \max_{\Phi} \int_0^1 \int -\left(\frac{\partial \Phi}{\partial t} + H(\nabla \Phi)\right) \rho(x,t)\, dxdt + \int \Phi(x,1)\rho_b(x) - \Phi(x,0)\rho_a(x)\, dx. \tag{3.26}$$

3.3.2 Simplification via geodesic pushforward map

Notice that both variables $\rho(\cdot,t)$ and $\Phi(\cdot,t)$ are time-varying functions in the above saddle point problem (3.26). On one hand this is in an integral form thus also a weak form of OT problem that may accommodate general cost L. On the other hand, (3.26) requires to optimize in a large space of time-varying functions, which may increase the computational cost as well as the chance of falling into local optimas. We reduce the search space by leveraging the the following geodesic to facilitate our training process:

Theorem 3.3.1. *Suppose* $\{\boldsymbol{X}_t^*\}_{t=0}^1$ *is the trajectory obeying the optimal vector field of* (3.9) *with strictly convex Lagrangian* L*, then* $\frac{d^2 \boldsymbol{X}_t^*}{dt^2} = 0$*,*

where we utilize the property of optimal transporting trajectory, if L is strictly convex.

This result was discussed by [51] for 2-Wasserstein case ($L(\cdot) = \frac{|\cdot|^2}{2}$). The more general result was presented in Theorem 5.5 of [15]. This theorem illustrates that, from the particle point of view, the optimal transport process can be treated as a pushforward operation along the geodesics (straight lines) in $\mathbb{R}^d$. To be more precise, we denote $\{\rho^*(\cdot,t)\}_{t=0}^1$ as the optimal solution to (3.7). Denote $\{v^*(\cdot,t)\}_{t=0}^1$ as the optimal vector field in (3.9). Then $\{\boldsymbol{X}_t^*\}_{t=0}^1$ solves $\frac{d}{dt}\boldsymbol{X}_t^* = v^*(\boldsymbol{X}_t,t)$. Since $\frac{d^2}{dt^2}\boldsymbol{X}_t^* = 0$, this implies $\boldsymbol{X}_t^* = \boldsymbol{X}_0^* + tv^*(\boldsymbol{X}_0^*,0)$, $t \in [0,1]$. Finally, due to the equivalence between (3.7) and (3.9), we are able to verify that $\boldsymbol{X}_t^* \sim \rho^*(\cdot,t)$, which yields $\boldsymbol{X}_0^* + tv^*(\boldsymbol{X}_0^*,0) \sim \rho^*(\cdot,t)$. Since $\boldsymbol{X}_0^*$ follows the distribution with density ρ_a, this leads to the following relation between optimal density $\rho^*(\cdot,t)$ and optimal vector field $v^*(\cdot,0)$ at $t=0$

$$\rho^*(\cdot,t) = (\mathrm{Id} + tv^*(\cdot,0))_\sharp \rho_a. \tag{3.27}$$

Equation (3.27) leads to the fact that the optimal $\rho^*(\cdot,t)$ can be obtained by pushforwarding the initial distribution ρ_a with the initial direction $v^*(\cdot,0)$ along certain straight lines. This observation motivates us to narrow the search space of $\{\rho_t\}_{t=0}^1$ on $\mathcal{P}_{\text{restrict}}$

$$\{\ \{\rho(\cdot,t)\}_{t=0}^1 \mid\ \rho(\cdot,t) = (\mathrm{Id} + tF)_\sharp \rho_a \ \text{ for } t \in [0,1]\ \ \},$$

where $F \in \mathbb{R}^d$. Combining above discussions we propose the scheme $\min_{\rho\in\mathcal{P}_{\text{restrict}}} \max_\Phi \hat{\mathfrak{L}}(\rho,\Phi)$. Specifically, considering ρ is uniquely determined by F, thus we reformulate our scheme as

$$\min_F \max_\Phi \mathcal{L}(F,\Phi), \quad \mathcal{L}(F,\Phi) = \hat{\mathfrak{L}}((\mathrm{Id}+tF)_\sharp\rho_a, \Phi). \tag{3.28}$$

We have the following theoretical property for scheme (3.28).

Theorem 3.3.2. *Denote the solution* (3.10) *as* $\rho^*(x,t)$ *and* $\Phi^*(x,t)$*, and set* $\Phi_0^*(\cdot) = \Phi^*(\cdot,0)$*. Assume* $\Phi^*(\cdot,t) \in C^2(\mathbb{R}^d)$*, then* $(\nabla L^{-1}(\nabla\Phi_0^*), \Phi^*)$ *is a critical point to the func-*

tional $\mathcal{L}$, i.e.

$$\frac{\partial \mathcal{L}}{\partial F}(\nabla L^{-1}(\nabla \Phi_0^*), \Phi^*) = 0, \quad \frac{\partial \mathcal{L}}{\partial \psi}(\nabla L^{-1}(\nabla \Phi_0^*), \Phi^*) = 0.$$

Furthermore, $\mathcal{L}(\nabla L^{-1}(\nabla \Phi_0^*), \Phi^*) = W_{Dym}(\rho_a, \rho_b)$.

Theorem 3.3.2 shows that the optimal solution of Dynamical OT is also a critical point of the functional used in our saddle scheme optimization (3.28). The optimal value of (3.28) is exactly the optimal transport distance. Before the proof of this Theorem, there are some Lemmas to be stated first.

Let us denote $F : \mathbb{R}^d \to \mathbb{R}^d$ as the transporting vector field. Recall that we are computing for the Wasserstein geodesic interpolating ρ_a and ρ_b. We denote $\hat{\rho}_t = (I + tF)_\sharp \rho_a$. Then we introduce the following functional of F and Φ:

$$\begin{aligned}
\mathcal{L}(F, \Phi) &= \int_0^1 \int \left(-\frac{\partial \Phi(x,t)}{\partial t} - H(\nabla \Phi(x,t)) \right) \hat{\rho}(x,t)\, dxdt \\
&\quad + \int \Phi(x,1)\rho_b(x) - \Phi(x,0)\rho_a(x)\, dx \qquad (3.29) \\
&= \int_0^1 \int \left(-\frac{\partial \Phi(x+tF(x),t)}{\partial t} - H(\nabla \Phi(x+tF(x),t)) \right) \rho_a(x,t)\, dxdt \\
&\quad + \int \Phi(x,1)\rho_b(x) - \Phi(x,0)\rho_a(x)\, dx
\end{aligned}$$

As mentioned in (3.28), our numerical method is to solve the following saddle point problem

$$\min_F \max_\Phi \mathcal{L}(F, \Phi) \qquad (3.30)$$

As stated in Theorem 3.3.2, the optimal solution obtained from dynamical OT problem (Brenier-Benamou formulation) (3.7) is a critical point to the functional $\mathcal{L}(F, \Phi)$.

Lemma 3.3.3. *Given a distribution with density ρ defined on $\mathbb{R}^d$, consider vector field $F : \mathbb{R}^d \to \mathbb{R}^d$. Define time-varying density $\{\rho(\cdot, t)\}_{t\in[0,1]}$ as $\rho(\cdot, t) = (Id + tF)_\sharp \rho_0$.*

Suppose for a given $f \in C^1(\mathbb{R}^d)$, $f(x)\rho(x,t)$ is integrable on $\mathbb{R}^d$. Then

$$\int f(x)\frac{\partial}{\partial t}\rho(x,t) = \int \nabla f(x+tF(x)) \cdot F(x)\ \rho_a(x)\ dx$$

Proof. We have

$$\begin{aligned}\int f(x)\frac{\partial}{\partial t}\rho(x,t) = \frac{d}{dt}\left(\int f(x)\rho(x,t)\ dx\right) =& \frac{d}{dt}\left(\int f(x+tF(x))\rho_a(x)\ dx\right) \\ =& \int \nabla f(x+F(x)) \cdot F(x)\ \rho_a(x)\ dx\end{aligned}$$

□

Lemma 3.3.4. *Suppose $\Phi^*(x,t)$ is solved from* (3.10) *in the section with initial condition $\Phi^*(\cdot,0) = \Phi_0^*(\cdot)$, we further assume $\Phi^*(\cdot,t) \in C^2(\mathbb{R}^d)$. Then we have*

$$\nabla\Phi^*(x + t\ \nabla L^{-1}(\nabla\Phi_0^*(x)),\ t) = \nabla\Phi_0^*(x). \tag{3.31}$$

Proof. Now consider Hamilton-Jacobi equation of (3.10) in the section:

$$\frac{\partial\Phi^*(y,t)}{\partial t} + H(\nabla\Phi^*(y,t)) = 0 \quad \Phi^*(\cdot,0) = \Phi_0^*.$$

We take gradient with respect to x on both sides, we have

$$\frac{\partial}{\partial t}(\nabla\Phi^*(y,t)) + \nabla^2\Phi^*(y,t)\nabla H(\nabla\Phi^*(y,t)) = 0. \tag{3.32}$$

Let us denote $T_t(x) = x + t\nabla H(\nabla\Phi_0^*(x))$ for simplicity. We now compute

$$\frac{d}{dt}\nabla\Phi^*(T_t(x),t) = \frac{\partial}{\partial t}\nabla\Phi^*(T_t(x),t) + \nabla^2\Phi^*(T_t(x),t)\nabla H(\nabla\Phi_0^*(x))$$

By plugging $y = T_t(x)$ into (3.32), we are able to verify $\frac{d}{dt}\nabla\Phi^*(T_t(x), t) = 0$. Thus

$$\nabla\Phi^*(T_t(x), t) = \nabla\Phi^*(T_0(x), 0) = \nabla\Phi_0^*(x) \quad \text{for } t \in [0, 1] \tag{3.33}$$

Recall H defined before, we can verify that $\nabla H = \nabla L^{-1}$. Thus (3.33) leads to

$$\nabla\Phi^*(x + t\,\nabla L^{-1}(\nabla\Phi_0^*(x)),\ t) = \nabla\Phi_0^*(x).$$

□

Lemma 3.3.5. *Suppose* $\Phi^*(x, t)$ *is solved from* (3.10) *with initial condition* $\Phi^*(\cdot, 0) = \Phi_0^*(\cdot)$*, we further assume* $\Phi^*(\cdot, t) \in C^2(\mathbb{R}^d)$*. Now denote* $\hat{\rho}(\cdot, t) = (Id + t\nabla L^{-1}(\nabla\Phi_0^*))_\sharp\rho_a$*. Then* $\hat{\rho}(\cdot, t)$ *solves*

$$\frac{\partial\hat{\rho}(x, t)}{\partial t} + \nabla\cdot(\hat{\rho}(x, t)\nabla L^{-1}(\nabla\Phi^*(x, t))) = 0.$$

Proof. For arbitrary $f \in C_0^\infty(\mathbb{R}^d)$, we consider:

$$\begin{aligned}
&\int f(x)\left(\frac{\partial\hat{\rho}(x, t)}{\partial t} + \nabla\cdot(\hat{\rho}(x, t)\nabla L^{-1}(\nabla\Phi^*(x, t)))\right)\,dx \\
=&\int f(x)\frac{\partial\hat{\rho}(x, t)}{\partial t}\,dx - \int \nabla f(x)\cdot\nabla L^{-1}(\nabla\Phi^*(x, t))\hat{\rho}(x, t)\,dx
\end{aligned}$$

By Lemma 3.3.3, the first term equals

$$\int \nabla f(x + t\nabla L^{-1}(\nabla\Phi_0^*(x)))\cdot\nabla L^{-1}(\nabla\Phi_0^*(x))\ \rho_a(x)\,dx \tag{3.34}$$

The second term equals

$$\int \nabla f(x + t\nabla L^{-1}(\nabla\Phi_0^*(x)))\cdot\nabla L^{-1}(\nabla\Phi^*(x + t\nabla L^{-1}(\nabla\Phi_0^*(x)), t))\ \rho_a(x)\,dx \tag{3.35}$$

Using Lemma 3.3.4, we know the integrals (3.34) and (3.35) are the same. Thus we have

$$\int f(x)\left(\frac{\partial\hat{\rho}(x,t)}{\partial t}+\nabla\cdot(\hat{\rho}(x,t)\nabla L^{-1}(\nabla\Phi^*(x,t)))\right)\,dx=0\quad\forall\ f\in C_0^\infty(\mathbb{R}^d).$$

This leads to our result. □

Lemma 3.3.6. *Suppose* $\Phi^*(x,t)$ *is solved from* (3.10) *with initial condition* $\Phi^*(\cdot,0)=\Phi_0^*(\cdot)$*, then*

$$W_{Dym}(\rho_a,\rho_b)=\int L(\nabla L^{-1}(\nabla\Phi_0^*(x)))\rho_a(x)\,dx. \tag{3.36}$$

Proof. Consider particle dynamical OT with its optimal solution $v^*(x,t)=\nabla L^{-1}(\nabla\Phi^*(x,t))$ as stated in (3.11) before. Recall Theorem 1 stating that the optimal plan is transporting each particle $\boldsymbol{X}_t$ along straight lines with constant velocity $v^*(\boldsymbol{X}_0,0)=\nabla L^{-1}(\nabla\Phi_0^*(\boldsymbol{X}_0))$, i.e. $v^*(\boldsymbol{X}_t,t)=v^*(\boldsymbol{X}_0,0)$ for any $t\in[0,1]$. Combining these, we have

$$W_{\text{Dym OT}}(\rho_a,\rho_b)=\int_0^1\mathbb{E}\,L(v^*(\boldsymbol{X}_t,t))\,dt=\mathbb{E}\left(\int_0^1 L(v^*(\boldsymbol{X}_t,t))\,dt\right)=\mathbb{E}\,L(\nabla L^{-1}(\nabla\Phi_0^*(\boldsymbol{X}_0))).$$

Notice that we require $\boldsymbol{X}_0\sim\rho_a$. This will lead to (3.36). □

Finally we provide the proof of Theorem 3.3.2.

Proof. Since we have assumed $\Phi^*(\cdot,t)\in C^2(\mathbb{R}^d)$, we restrict our $\Phi\in C^2(\mathbb{R}^d)$ as well. We first rewrite $\mathcal{L}(F,\Phi)$ by using integration by parts as:

$$\int_0^1\int\Phi(x,t)\frac{\partial\hat{\rho}(x,t)}{\partial t}-H(\nabla\Phi(x,t))\hat{\rho}(x,t)\,dxdt+\int\Phi(x,1)(\rho_a(x)-\hat{\rho}(x,1))\,dx. \tag{3.37}$$

By Lemma 3.3.3, (3.37) can be written as

$$\begin{aligned}\mathcal{L}(F,\Phi)=&\int_0^1\int_{\mathbb{R}^d}[\nabla\Phi(x+tF(x),t)\cdot F(x)-H(\nabla\Phi(x+tF(x),t))]\rho_a(x)\,dxdt \\ &+\int\Phi(x,1)\rho_b(x)\,dx-\int\Phi(x+F(x),1)\rho_a(x)\,dx.\end{aligned} \tag{3.38}$$

Now based on (3.38) here, we are able to compute $\frac{\partial \mathcal{L}(F,\Phi)}{\partial F}(x)$ as

$$\begin{aligned}\frac{\partial \mathcal{L}(F,\Phi)}{\partial F} = &\int_0^1 t\nabla^2\Phi(x+tF(x),t)\cdot \underbrace{[F(x)-\nabla H(\nabla\Phi(x+tF(x),t))]}_{(A)}\rho_a(x)\,dt \\ &+ \underbrace{\left(\int_0^1 \nabla\Phi(x+tF(x),t)\,dt - \nabla\Phi(x+F(x),1)\right)}_{(B)}\rho_a(x).\end{aligned} \tag{3.39}$$

Now we plug $F=\nabla L^{-1}(\nabla\Phi_0^*)$, $\Phi=\Phi^*$ into (3.39), by Lemma 3.3.4, we have

$$\nabla\Phi^*(x+t\,\nabla L^{-1}(\nabla\Phi_0^*(x)),\ t) = \nabla\Phi_0^*(x). \tag{3.40}$$

Then using (3.40) and recall that $\nabla H = \nabla L^{-1}$, one can verify that $(A)=0$, similarly, for (B), we have $\nabla\Phi(x+tF(x),t)=\nabla\Phi_0^*$ for all $t\in[0,1]$. Thus $(B)=0$ and we are able to verify $\frac{\partial\mathcal{L}}{\partial F}(\nabla L^{-1}(\nabla\Phi_0^*),\Phi^*)=0$.

On the other hand, we can compute $\frac{\partial\mathcal{L}(F,\Phi)}{\partial\Phi}(x,t)$ as

$$\frac{\delta\mathcal{L}(F,\Phi)}{\delta\Phi}(x,t) = \underbrace{\left[\frac{\partial\hat{\rho}(x,t)}{\partial t}+\nabla\cdot(\hat{\rho}(x,t)\nabla H(\nabla\Phi(x,t)))\right]}_{(C)} + \underbrace{[\rho_b(x)-\hat{\rho}(x,1)]}_{(D)}\delta_1(t)$$

Now by Lemma 3.3.5, we know $(C)=0$. Furthermore, since Φ^* solves Dynamical OT problem associated to the optimal transport problem between ρ_a and ρ_b, by (3.13), we have $\hat{\rho}(x,1)=(Id+\nabla\Phi_0^*)_\sharp\rho_a=\rho_b$, this verifies $(D)=0$. Thus, we are able to verify $\frac{\partial\mathcal{L}(F,\Phi)}{\partial\Phi}(\nabla L^{-1}(\nabla\Phi_0^*),\Phi^*)=0$.

At last, we plug $F=\nabla L^{-1}(\nabla\Phi_0^*)$, $\Phi=\Phi^*$ in (3.38) to obtain:

$$\begin{aligned}\mathcal{L}(\nabla L^{-1}(\nabla\Phi_0^*),\Phi^*) &= \int_0^1\int \nabla\Phi_0^*(x)\cdot\nabla L^{-1}(\nabla\Phi_0^*(x)) - H(\nabla\Phi_0^*(x))\,\rho_a(x)\,dxdt \\ &= \int L(\nabla L^{-1}(\nabla\Phi_0^*(x)))\rho_a(x)\,dx.\end{aligned}$$

Now by Lemma 3.3.6, we have verified $\mathcal{L}(\nabla L^{-1}(\nabla\Phi_0^*), \Phi^*) = W_{\text{Dym OT}}(\rho_a, \rho_b)$. $\square$

3.3.3 Bidirectional dynamical formulation

We would like to improve the stability and avoid local traps during the training process by proposing a *bidirectional* scheme, which consider the symmetric status of ρ_a and ρ_b in (3.1). First of all for two OT problems

$$\min_F \max_{\Phi_F} \mathcal{L}^{ab}(F, \Phi_F), \quad \min_G \max_{\Phi_G} \mathcal{L}^{ba}(G, \Phi_G),$$

where $\mathcal{L}^{ab}$ is defined in (3.28), $\mathcal{L}^{ba}$ is defined by switching ρ_a and ρ_b in (3.28). Clearly, the first one is for $W(\rho_a, \rho_b)$ and the second one is for $W(\rho_b, \rho_a)$. At optima, the vector fields F and G are transport vectors in the opposite directions. At a specific point $x \in \mathbb{R}^d$, moving along straight line in the direction F ends up at $x + F(x)$. The direction of G at $x + F(x)$ should point to the opposite direction of $F(x)$, which leads to $G(x + F(x)) = -F(x)$. Similarly, we also have $F(x + G(x)) = -G(x)$. With above two conditions we add two constraints for F and G to facilitate our training:

$$\mathcal{R}^{ab}(F, G) = \int |G(x + F(x)) + F(x)|^2 \, \rho_a(x) \, dx,$$
$$\mathcal{R}^{ba}(F, G) = \int |F(x + G(x)) + G(x)|^2 \rho_b(x) \, dx.$$

Our final saddle-point problem becomes

$$\min_{F,G} \max_{\Phi_F, \Phi_G} \mathcal{L}^{ab}(F, \Phi_F) + \mathcal{L}^{ba}(G, \Phi_G) + \lambda(\mathcal{R}^{ab}(F, G) + \mathcal{R}^{ba}(F, G)), \tag{3.41}$$

where λ is a tunable coefficient of our constraint terms.

3.3.4 Overview of the algorithm

To solve (3.41), we propose an algorithm that is summarized in the following steps.

- **Preconditioning** We can apply preconditioning techniques to 2-Wasserstein cases in order to make our computation more efficient.

- **Main Algorithm** We set $F_{\theta_1}, G_{\theta_2}$ and $\Phi^F_{\omega_1}, \Phi^G_{\omega_2}$ as fully connected neural networks and optimize over their parameters ω_1, ω_2 and θ_1, θ_2 alternatively via stochastic gradient ascend and descend.

- **Stopping Criteria** When computed F (or G) is close to the optimal solution, the Wasserstein distance $W(\rho_a, \rho_b)$ (or $W(\rho_b, \rho_a)$) can be approximated by

$$\widehat{W}^{ab} = \int L(F(x))\ \rho_a(x)\ dx,\ \widehat{W}^{ba} = \int L(G(x))\ \rho_b(x)\ dx.$$

  For a chosen threshold $\epsilon > 0$, we treat $|\widehat{W}^{ab} - \widehat{W}^{ba}| < \epsilon$ as the stopping criteria of our algorithm.

It's worth mentioning that we can apply preconditioning technique under the 2-Wasserstein cases, i.e., $L(\cdot) = \frac{|\cdot|^2}{2}$. When the support of distributions ρ_a and ρ_b are far away from each other, the computational process might get much more sensitive with respect to vector field F. In order to deal with this situation, we consider preconditioning to our initial distribution ρ_a. In our implementation, we treat $P : \mathbb{R}^d \to \mathbb{R}^d$ as our preconditioning map. We fix its structure as $P(x) = \sigma x + \mu$ with $\sigma \in \mathbb{R}_+,\ \mu \in \mathbb{R}^d$. Such preconditioning process can be treated as an operation aiming at relocating and rescaling the initial distribution ρ_a so that the support of $P_\sharp \rho_a$ matches with the support of ρ_b in a better way, which in turn facilitates the training process of our OT problem. A similar technique is also carried out by [52].

Let us denote the optimal vector field of OT problem between $P_\sharp \rho_a$ and ρ_b as $\nabla\widehat{\Phi}_0$, then for the vector field $F^*(x) = \nabla\widehat{\Phi}_0 \circ P(x) + P(x) - x$, the following theorem guarantees the optimality of F^*.

Theorem 3.3.7. *Suppose $L(\cdot) = \frac{|\cdot|^2}{2}$. We define the map $P(x) = \sigma x + \mu$ with $\sigma \in \mathbb{R}_+, \mu \in \mathbb{R}^d$. Recall* (3.11)*, we denote $v(x,t) = \nabla\widehat{\Phi}(x,t)$ as the optimal solution to dynamical OT*

problem (3.7) *from* $P_\sharp\rho_a$ *to* ρ_b*. We set* $\widehat{\Phi}_0 = \widehat{\Phi}(\cdot, 0)$*. Furthermore, we denote* $v(x,t) = \nabla\Phi^*(x,t)$ *as the optimal solution to dynamical OT problem* (3.7) *from* ρ_a *to* ρ_b*, and set* $\Phi_0^* = \Phi^*(\cdot, 0)$*. Then we have*

$$\begin{aligned}\nabla\Phi_0^*(x) &= \nabla\widehat{\Phi}_0 \circ P(x) + P(x) - x,\\ \Phi_0^*(x) &= \frac{1}{\sigma}\widehat{\Phi}_0(\sigma x + \mu) + \frac{\sigma - 1}{2}|x|^2 + \mu^T x + \textit{Const}.\end{aligned}$$

This theorem indicates that our constructed F^* is exactly the optimal transport field $\nabla\Phi_0^*$ for the original OT problem from ρ_a to ρ_b.

Proof. According to (3.13) we have

$$(\mathrm{Id} + \nabla\widehat{\Phi}_0)_\sharp(P_\sharp\rho_a) = \rho_b$$

This yields

$$(P + \nabla\Phi_0 \circ P)_\sharp\rho_a = \rho_b$$

We rewrite this as

$$(\mathrm{Id} + \nabla\widehat{\Phi}_0 \circ P + P - \mathrm{Id})_\sharp\rho_a = \rho_b \tag{3.42}$$

We denote $u(x) = \frac{1}{\sigma}\widehat{\Phi}_0(\sigma x + \mu) + \frac{\sigma-1}{2}|x|^2 + \mu^T x$.

Then we can directly verify that

$$\nabla u(x) = \nabla\widehat{\Phi}_0(\sigma x + \mu) + (\sigma x + \mu) - x = \nabla\widehat{\Phi}_0 \circ P(x) + P(x) - x$$

Plug this into (3.42) above we get:

$$(\mathrm{Id} + \nabla u)_\sharp\rho_a = \rho_b$$

Using the uniqueness of the solution to Monge-Ampere equation, we have $\Phi_0^* = u + \mathrm{Const}$,

or equivalently, $\nabla\Phi_0^*(x) = \nabla u(x) = \nabla\widehat{\Phi}_0 \circ P(x) + P(x) - x$ □

Our computation procedure is summarized in Algorithm 1. We set $F_{\theta_1}, G_{\theta_2}$ and $\Phi^F_{\omega_1}, \Phi^G_{\omega_2}$ as fully connected neural networks and optimize over their parameters.

Algorithm 1 Computing Wasserstein geodesic from ρ_a to ρ_b via bidirectional scheme (3.41) and preconditioning

1: Choose our preconditioning map $P(x) = \sigma x + \mu$. Denote $\hat{\rho}_a = P_\sharp \rho_a$ (This step is only applicable for 2-Wasserstein case. If we do not need preconditioning, we treat $P = \text{Id}$.)
2: Set up the threshold $\epsilon > 0$ as the stopping criteria
3: Initialize $F_{\theta_1}, G_{\theta_2}, \Phi^F_{\omega_1}, \Phi^G_{\omega_2}$
4: **for** $F_{\theta_1}, G_{\theta_2}$ steps **do**
5: Sample $\{(z_k^a, t_k^a)\}_{k=1}^N$ from $\hat{\rho}_a \otimes U(0,1)$ and $\{(z_k^b, t_k^b)\}_{k=1}^N$ from $\rho_b \otimes U(0,1)$;
6: Set $x_k^a = z_k^a + t_k^a F_{\theta_1}(z_k^a)$, $x_k^b = z_k^b + t_k^b G_{\theta_2}(z_k^b)$;
7: Sample $\{w_k^a\}_{k=1}^M$ from $\hat{\rho}_a$ and $\{w_k^b\}$ from ρ_b;
8: **for** $\Phi^F_{\omega_1}, \Phi^G_{\omega_2}$ steps **do**
9: Update (via gradient ascent) $\Phi^F_{\omega_1}, \Phi^G_{\omega_2}$ by:

$$\nabla_{\omega_1,\omega_2}(\mathcal{L}^{ab}(\Phi^F_{\omega_1}) + \mathcal{L}^{ba}(\Phi^G_{\omega_2}))$$

10: **end for**
11: Sample $\{\xi_k^a\}_{k=1}^K$ from $\hat{\rho}_a$ and $\{\xi_k^b\}_{k=1}^K$ from ρ_b
12: Update (grad descent) $F_{\theta_1}, G_{\theta_2}$ by:

$$\nabla_{\theta_1,\theta_2}(\mathcal{L}^{ab}(\Phi^F_{\omega_1}) + \mathcal{L}^{ba}(\Phi^G_{\omega_2}) + \mathcal{K}(F_{\theta_1}, G_{\theta_2}))$$

13: Whenever $|\widehat{W}^{ab} - \widehat{W}^{ba}| < \epsilon$, skip out of the loop.
14: **end for**
15: Set $F^* = F_{\theta_1} \circ P + P - \text{Id}$ and $G^* = G_{\theta_2} \circ P + P - \text{Id}$.
16: Wasserstein geodesic from ρ_a to ρ_b is given by $\{(\text{Id} + tF_{\theta_1})_\sharp \rho_a\}$; Wasserstein geodesic from ρ_b to ρ_a is given by $\{(\text{Id} + tG_{\theta_2})_\sharp \rho_b\}$.

Remark 3. *In Algorithm 1, we need to sample points $\{z_k^a\}$ from the distribution $\hat{\rho}_a = P_\sharp \rho_a$. To achieve this, we first sample $\{u_k\}$ from ρ_a. Then $\{P(u_k)\}$ are our desired samples from $\hat{\rho}_a$.*

In Algorithm 1 we define

$$\mathcal{L}^{ab}(\Phi^F_{\omega_1}) = -\frac{1}{N}\sum_{k=1}^{N}\left[\frac{\partial}{\partial t}\Phi^F_{\omega_1}(x^a_k, t^a_k) + H(\nabla\Phi^F_{\omega_1}(x^a_k, t^a_k))\right] + \frac{1}{M}\sum_{k=1}^{M}(\Phi^F_{\omega_1}(w^b_k, 1) - \Phi^F_{\omega_1}(w^a_k, 0)),$$

$$\mathcal{L}^{ba}(\Phi^G_{\omega_2}) = -\frac{1}{N}\sum_{k=1}^{N}\left[\frac{\partial}{\partial t}\Phi^G_{\omega_2}(x^b_k, t^b_k) + H(\nabla\Phi^G_{\omega_2}(x^b_k, t^b_k))\right] + \frac{1}{M}\sum_{k=1}^{M}(\Phi^G_{\omega_2}(w^b_k, 1) - \Phi^G_{\omega_2}(w^a_k, 0)),$$

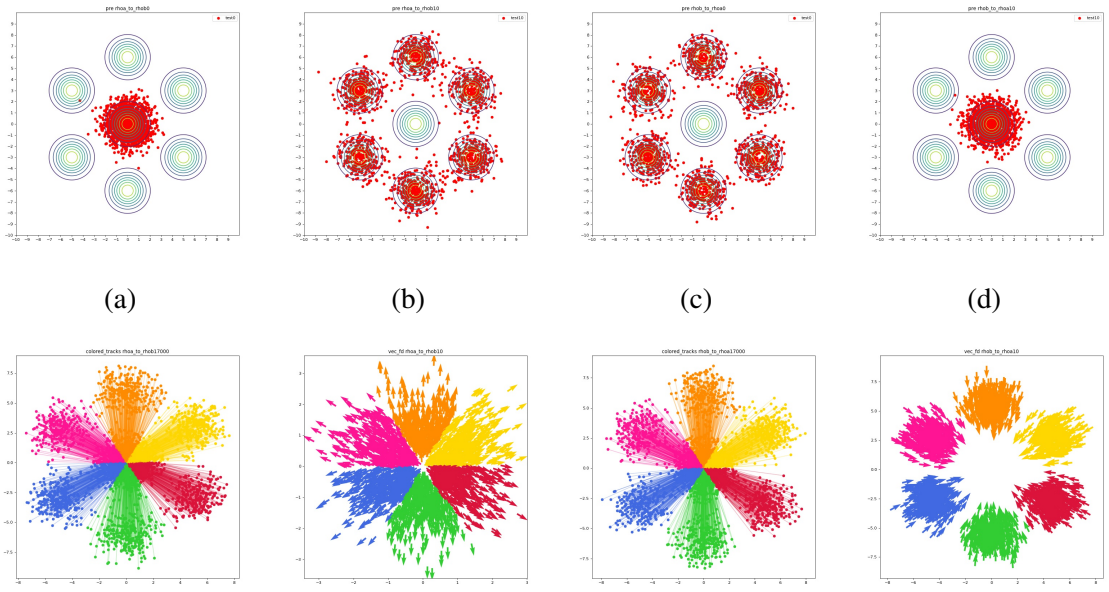
$$\mathcal{K}(F_{\theta_1}, G_{\theta_2}) = \frac{\lambda}{K}\sum_{k=1}^{K}|G_{\theta_2}(\xi^a_k + F_{\theta_1}(\xi^a_k)) + F_{\theta_1}(\xi^a_k)|^2 + \frac{\lambda}{K}\sum_{k=1}^{K}|F_{\theta_1}(\xi^b_k + G_{\theta_2}(\xi^b_k)) + G_{\theta_2}(\xi^b_k)|^2,$$

$$\widehat{W}^{ab} = \frac{1}{M}\sum_{k=1}^{M}L(F_{\theta_1}(w^a_k)), \quad \widehat{W}^{ba} = \frac{1}{M}\sum_{k=1}^{M}L(G_{\theta_2}(w^b_k)).$$

3.4 Experiments

Experiment Setup: We validate our algorithm through vast of synthetic data sets including different dimensional Gaussian cases. We also test our algorithm for realistic data sets including color transfer [53] and transportation between MNIST digits [54].

For low dimensional cases, namely, 2, 3, 5 and 10 dimensional cases, we set Φ_F, Φ_G and F, G as fully connected neural networks, where Φ_F, Φ_G have 6 hidden layers and F and G have 5 hidden layers. Each layer has 48 nodes, the activation function is chosen as Tanh. For high dimensional cases, namely, the data dimension is $28\times$ 28, we deal with MNIST handwritten digits data set, we adopt similar structures of neural networks, the only difference is that in each layer we extend the number of nodes from 48 to 512. In terms of training process, for all synthetic and realistic cases we use the Adam optimizer [55] with learning rate 10^{-4}. Notice that we are computing Wasserstein geodesic, namely, starting with an initial distribution ρ_{t_0}, in most cases we generate evolving distributions for next ten time steps, from $t_1 = 0.1$ to $t_{10} = 1$. The cost functions are chosen as $L(v) = |v|^{\frac{3}{2}}$ in Synthetic-2 and $L(v) = |v|^2$ in all other tests. We only show the final state of the generated distribution due to space limitation, more experiments and details are included in next section.



(a) (b) (c) (d)

(e) (f) (g) (h)

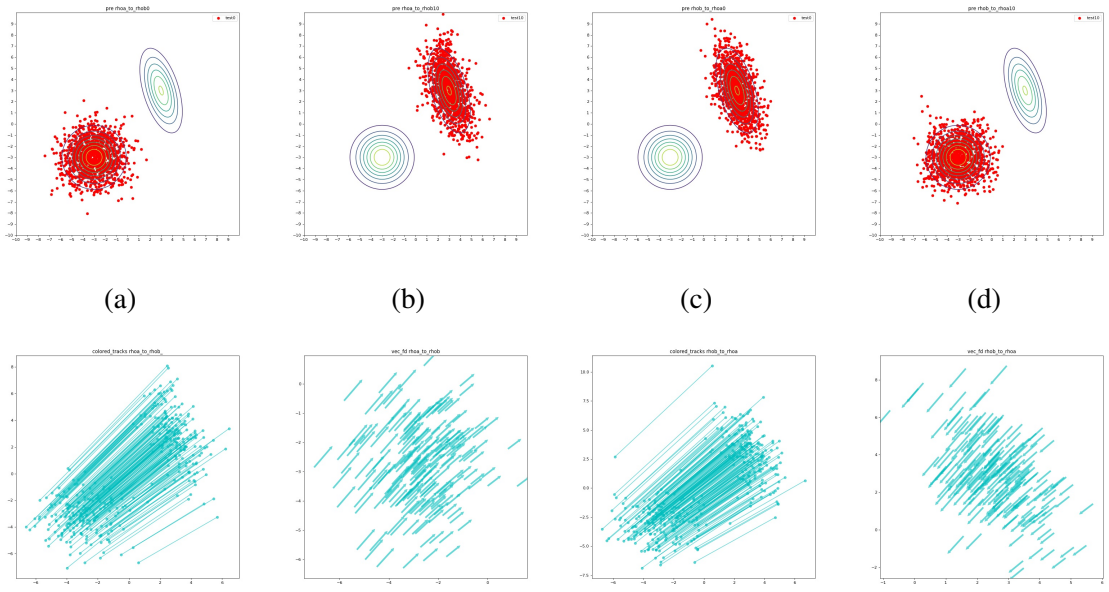
Figure 3.2: Syn-1. (a)(b) true ρ_a and generated ρ_b, (c)(d) true ρ_b and generated ρ_a, (e)(g) tracks of sample points from $\rho_a(\rho_b)$ to $\rho_b(\rho_a)$, (f)(h) vector fields from $\rho_a(\rho_b)$ to $\rho_b(\rho_a)$.

Synthetic-1: This is a 2-dimensional case, we set ρ_a as a standard Gaussian distribution $N(\mu_0, \Sigma_0)$ while ρ_b as six surrounding Gaussian distributions with the same Σ_0. In Figure 3.2, we show the generated distribution that follows ρ_b as well as the one that follows ρ_a. We also show the start-end tracks of points and vector field.

Synthetic-2: In this 5-dimensional case we treat ρ_a and ρ_b both as two Gaussian distributions. We show the results of two dimensional projection in Figure 3.3.

Training and Results: For synthetic data sets, in the training process we set the batch size $N_t = 2000$ and sample size for prediction $N_p = 1000$. From Figures 3.2, 3.3 we see that in all cases, within various dimensional settings, the generated samples closely follow the ground-truth distributions.

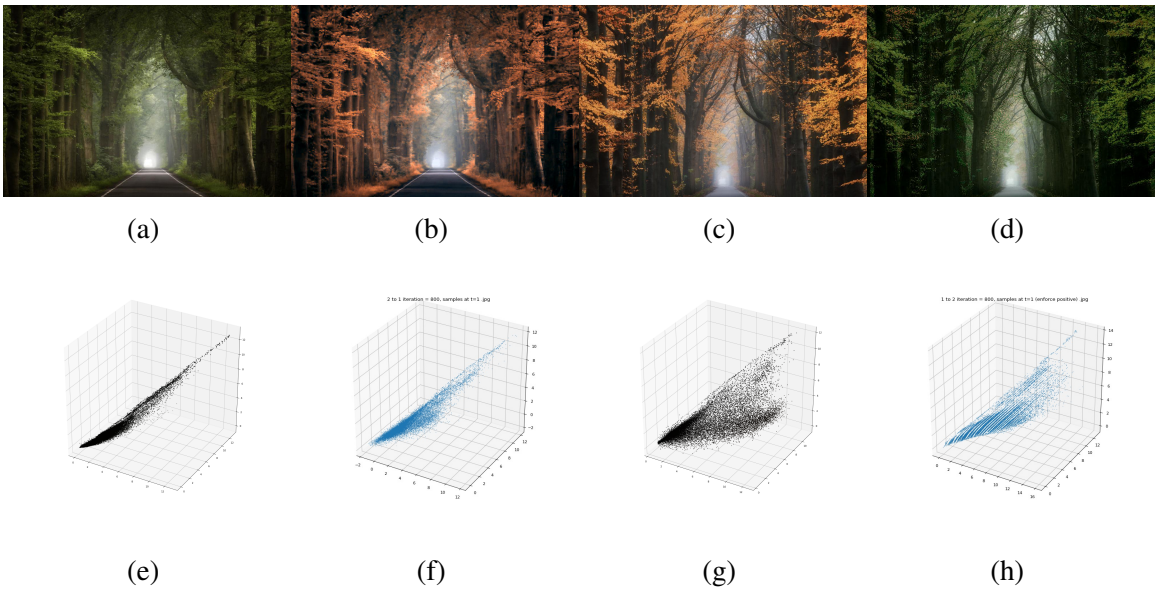
Realistic-1: Two given pictures describe the summer view(ρ_a) and autumn view(ρ_b) of a forest. We generate the autumn view starting with the summer view and generate the summer view starting with the autumn view. We also show the ground-truth and generated palette distributions in Figure 3.4.



(a) (b) (c) (d)

(e) (f) (g) (h)

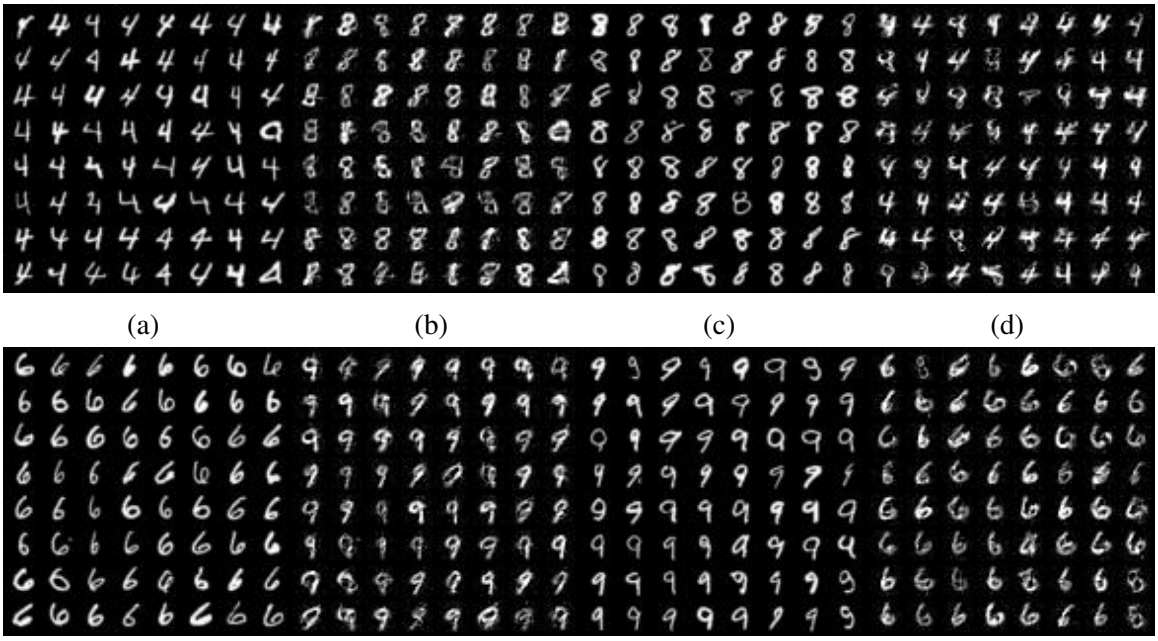
Figure 3.3: Syn-2. (a)(b) true ρ_a and generated ρ_b, (c)(d) true ρ_b and generated ρ_a, (e)(g) tracks of sample points from $\rho_a(\rho_b)$ to $\rho_b(\rho_a)$, (f)(h) vector fields from $\rho_a(\rho_b)$ to $\rho_b(\rho_a)$.



(a) (b) (c) (d)

(e) (f) (g) (h)

Figure 3.4: Real-1, Forest views. (a)(c) true summer(autumn) view, (b)(d) generated autumn(summer) view when true summer(autumn) view is given, (e)(g) true palette distribution of summer(autumn) view, (f)(h) generated palette distribution of summer(autumn) view.



(a) (b) (c) (d)

(e) (f) (g) (h)

Figure 3.5: Real-2, Digits transformation. (a)(b) true(generated) digit 4(8), (c)(d) true(generated) digit 8(4), (e)(f) true(generated) digit 6(9), (g)(h)true(generated) digit 9(6).

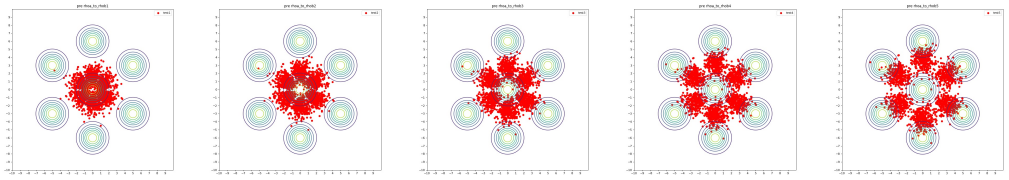
Realistic-2: We choose MNIST as our data set (28×28 dimensional) and study the Wasserstein mappings as well as geodesic between digit $0(\rho_a)$ and digit $1(\rho_b)$, digit $4(\rho_a)$ and digit $8(\rho_b)$, digit $6(\rho_a)$ and digit $9(\rho_b)$. Only partial results are presented in Figure 3.5 due to space limit.

Training and Results: For realistic-1 we set the batch size $N_t = 1000$, for realistic-2 in each iteration we take $N_t = 500$ pictures for training, especially for realistic-2 we also add small noise to samples during the training process. From Figure 3.4 we see that the generated autumn(summer) views are very close to the true autumn(summer) views, moreover, the similarity between true and generated palette distributions also demonstrate that our algorithm works well in these cases. In realistic-2 we study the Wasserstein geodesic and mappings in original space without any dimension reduction techniques. Our generated handwritten digits follow similar patterns with the ground truth. In next section we present our full experimental results.

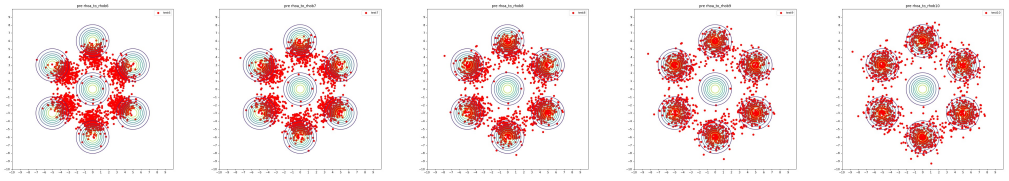
3.5 Complete Geodesics in Experiments

3.5.1 Synthetic Data

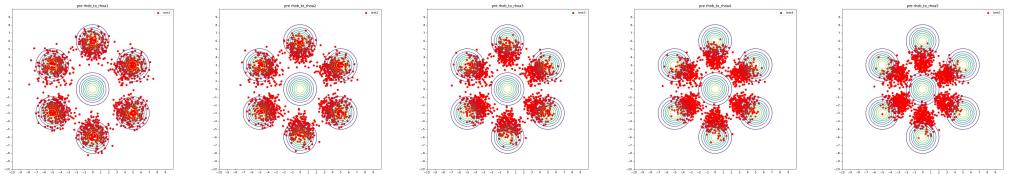
Syn-1: Wasserstein-2



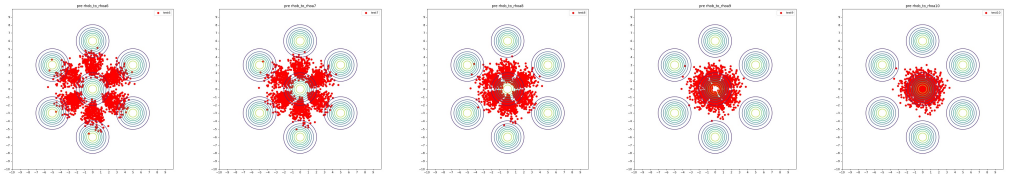
(a) ρ_a to ρ_b at t_1 (b) ρ_a to ρ_b at t_2 (c) ρ_a to ρ_b at t_3 (d) ρ_a to ρ_b at t_4 (e) ρ_a to ρ_b at t_5



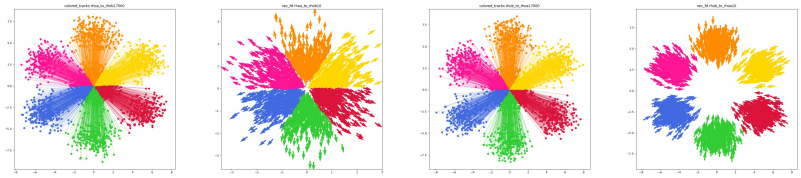
(f) ρ_a to ρ_b at t_6 (g) ρ_a to ρ_b at t_7 (h) ρ_a to ρ_b at t_8 (i) ρ_a to ρ_b at t_9 (j) ρ_a to ρ_b at t_{10}



(k) ρ_b to ρ_a at t_1 (l) ρ_b to ρ_a at t_2 (m) ρ_b to ρ_a at t_3 (n) ρ_b to ρ_a at t_4 (o) ρ_b to ρ_a at t_5



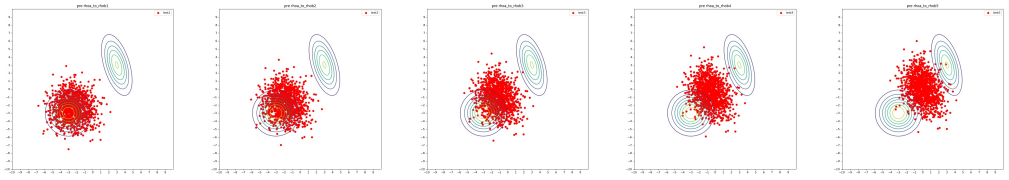
(p) ρ_b to ρ_a at t_6 (q) ρ_b to ρ_a at t_7 (r) ρ_b to ρ_a at t_8 (s) ρ_b to ρ_a at t_9 (t) ρ_b to ρ_a at t_{10}



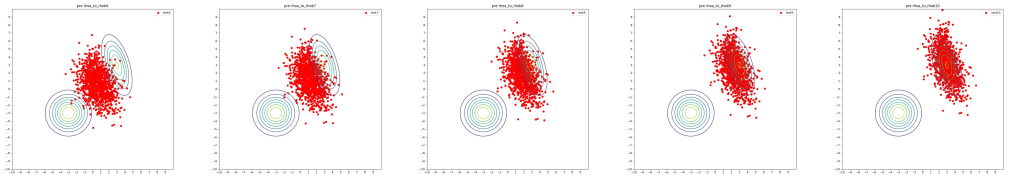
(u) traj ρ_a to ρ_b (v) vf ρ_a to ρ_b (w) traj ρ_b to ρ_a (x) vf ρ_b to ρ_a

Figure 3.6: 2-dimensional Gaussian to Gaussian

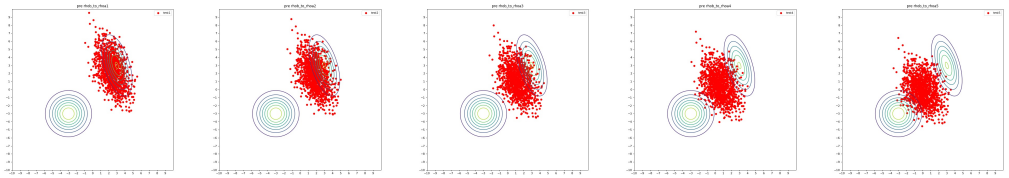
Syn-2: Wasserstein-1.5



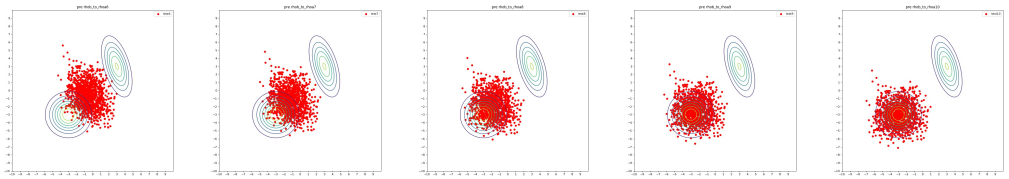
(a) ρ_a to ρ_b at t_1 (b) ρ_a to ρ_b at t_2 (c) ρ_a to ρ_b at t_3 (d) ρ_a to ρ_b at t_4 (e) ρ_a to ρ_b at t_5



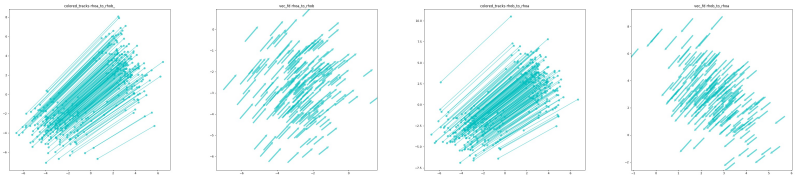
(f) ρ_a to ρ_b at t_6 (g) ρ_a to ρ_b at t_7 (h) ρ_a to ρ_b at t_8 (i) ρ_a to ρ_b at t_9 (j) ρ_a to ρ_b at t_{10}



(k) ρ_b to ρ_a at t_1 (l) ρ_b to ρ_a at t_2 (m) ρ_b to ρ_a at t_3 (n) ρ_b to ρ_a at t_4 (o) ρ_b to ρ_a at t_5



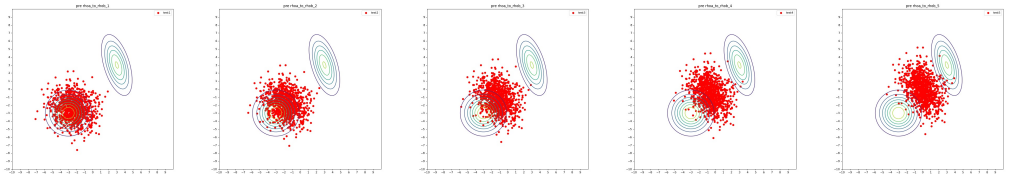
(p) ρ_b to ρ_a at t_6 (q) ρ_b to ρ_a at t_7 (r) ρ_b to ρ_a at t_8 (s) ρ_b to ρ_a at t_9 (t) ρ_b to ρ_a at t_{10}



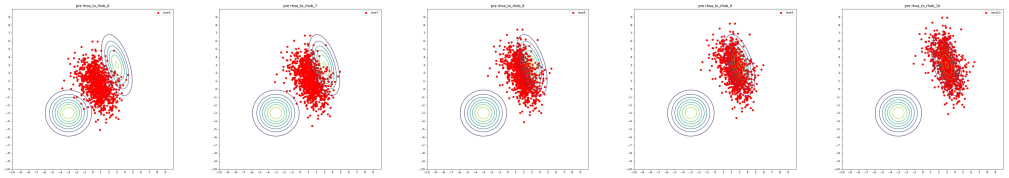
(u) traj ρ_a to ρ_b (v) vf ρ_a to ρ_b (w) traj ρ_b to ρ_a (x) vf ρ_b to ρ_a

Figure 3.7: 5-dimensional Gaussian to Gaussian

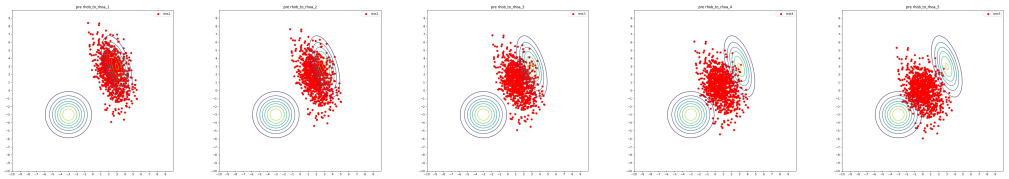
Syn-3: Wasserstein-2



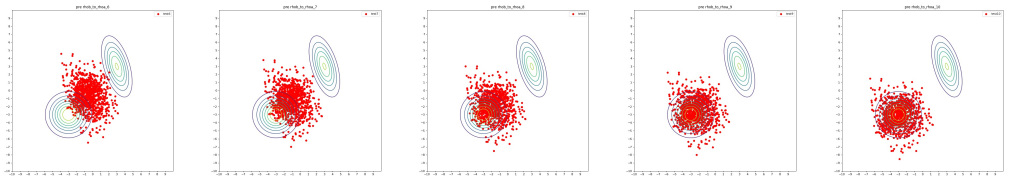
(a) ρ_a to ρ_b at t_1 (b) ρ_a to ρ_b at t_2 (c) ρ_a to ρ_b at t_3 (d) ρ_a to ρ_b at t_4 (e) ρ_a to ρ_b at t_5



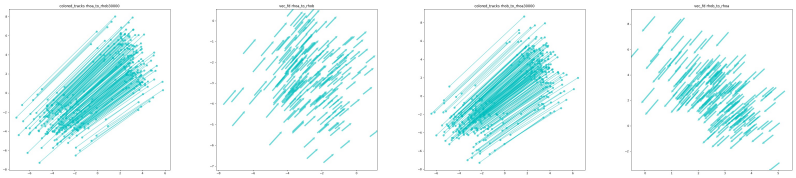
(f) ρ_a to ρ_b at t_6 (g) ρ_a to ρ_b at t_7 (h) ρ_a to ρ_b at t_8 (i) ρ_a to ρ_b at t_9 (j) ρ_a to ρ_b at t_{10}



(k) ρ_b to ρ_a at t_1 (l) ρ_b to ρ_a at t_2 (m) ρ_b to ρ_a at t_3 (n) ρ_b to ρ_a at t_4 (o) ρ_b to ρ_a at t_5



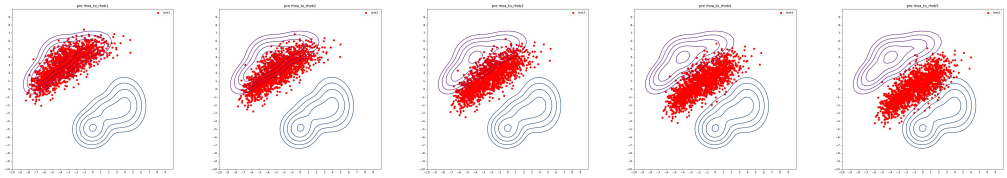
(p) ρ_b to ρ_a at t_6 (q) ρ_b to ρ_a at t_7 (r) ρ_b to ρ_a at t_8 (s) ρ_b to ρ_a at t_9 (t) ρ_b to ρ_a at t_{10}



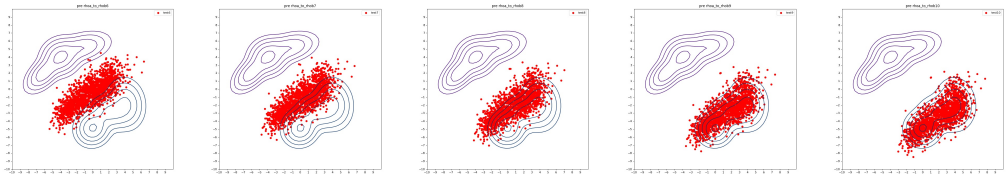
(u) traj ρ_a to ρ_b (v) vf ρ_a to ρ_b (w) traj ρ_b to ρ_a (x) vf ρ_b to ρ_a

Figure 3.8: 10-dimensional Gaussian to Gaussian

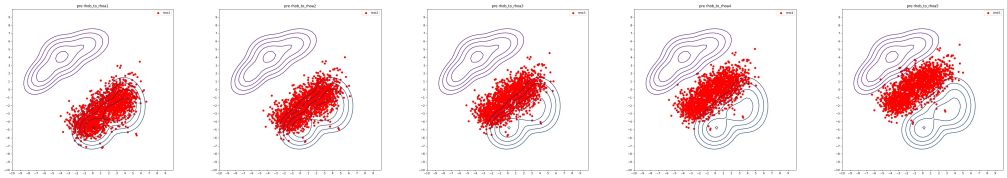
Syn-3: Wasserstein-2



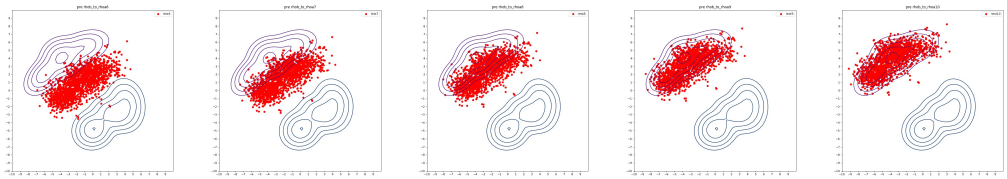
(a) ρ_a to ρ_b at t_1 (b) ρ_a to ρ_b at t_2 (c) ρ_a to ρ_b at t_3 (d) ρ_a to ρ_b at t_4 (e) ρ_a to ρ_b at t_5



(f) ρ_a to ρ_b at t_6 (g) ρ_a to ρ_b at t_7 (h) ρ_a to ρ_b at t_8 (i) ρ_a to ρ_b at t_9 (j) ρ_a to ρ_b at t_{10}



(k) ρ_b to ρ_a at t_1 (l) ρ_b to ρ_a at t_2 (m) ρ_b to ρ_a at t_3 (n) ρ_b to ρ_a at t_4 (o) ρ_b to ρ_a at t_5



(p) ρ_b to ρ_a at t_6 (q) ρ_b to ρ_a at t_7 (r) ρ_b to ρ_a at t_8 (s) ρ_b to ρ_a at t_9 (t) ρ_b to ρ_a at t_{10}

Figure 3.9: 10-dimensional Gaussian Mixture

3.5.2 Realistic Data

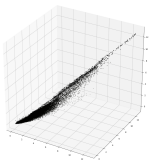
Real-1: Color Transfer of Forest View

(a) ρ_a to ρ_b at t_1 (b) ρ_a to ρ_b at t_2 (c) ρ_a to ρ_b at t_3 (d) ρ_a to ρ_b at t_4 (e) ρ_a to ρ_b at t_5

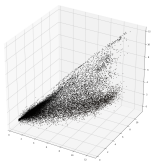
(f) ρ_a to ρ_b at t_6 (g) ρ_a to ρ_b at t_7 (h) ρ_a to ρ_b at t_8 (i) ρ_a to ρ_b at t_9 (j) ρ_a to ρ_b at t_{10}

(k) ρ_b to ρ_a at t_1 (l) ρ_b to ρ_a at t_2 (m) ρ_b to ρ_a at t_3 (n) ρ_b to ρ_a at t_4 (o) ρ_b to ρ_a at t_5

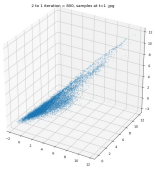
(p) ρ_b to ρ_a at t_6 (q) ρ_b to ρ_a at t_7 (r) ρ_b to ρ_a at t_8 (s) ρ_b to ρ_a at t_9 (t) ρ_b to ρ_a at t_{10}



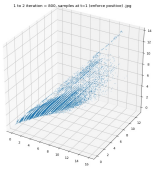
(u) Color dist of Summer View



(v) Color dist of Autumn View



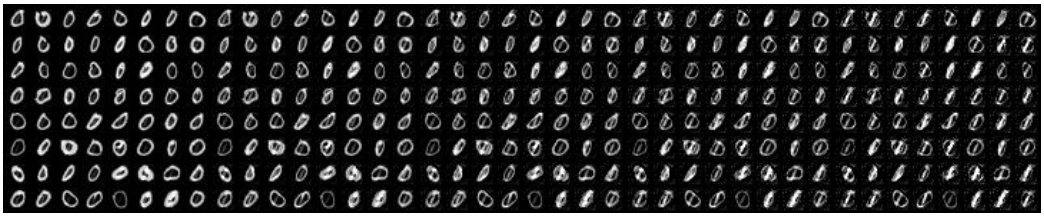
(w) Generated color dist of Summer View



(x) Generated color dist of Autumn View

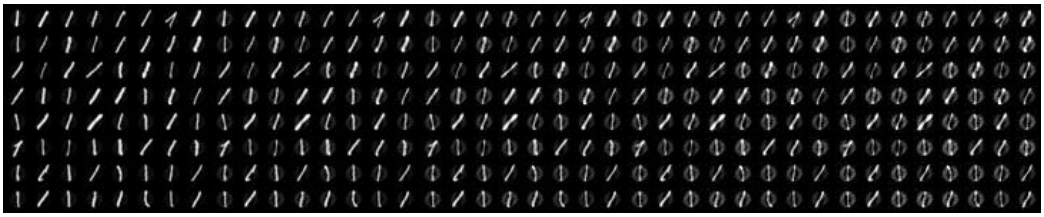
Figure 3.10: Forest summer view and autumn view

Real-2: MNIST

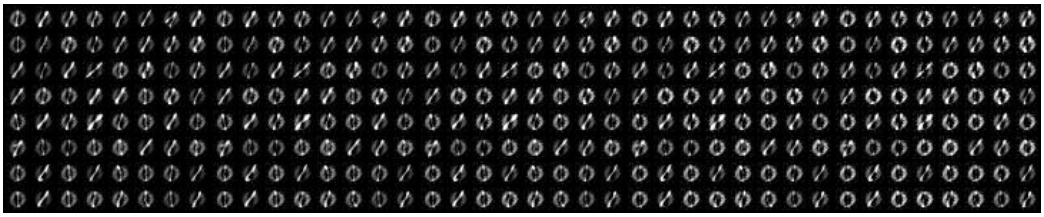


(a) ρ_a to ρ_b at t_1 (b) ρ_a to ρ_b at t_2 (c) ρ_a to ρ_b at t_3 (d) ρ_a to ρ_b at t_4 (e) ρ_a to ρ_b at t_5

(f) ρ_a to ρ_b at t_6 (g) ρ_a to ρ_b at t_7 (h) ρ_a to ρ_b at t_8 (i) ρ_a to ρ_b at t_9 (j) ρ_a to ρ_b at t_{10}



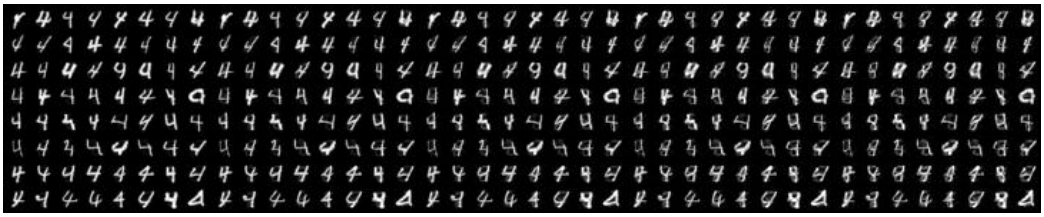
(k) ρ_b to ρ_a at t_1 (l) ρ_b to ρ_a at t_2 (m) ρ_b to ρ_a at t_3 (n) ρ_b to ρ_a at t_4 (o) ρ_b to ρ_a at t_5



(p) ρ_b to ρ_a at t_6 (q) ρ_b to ρ_a at t_7 (r) ρ_b to ρ_a at t_8 (s) ρ_b to ρ_a at t_9 (t) ρ_b to ρ_a at t_{10}

Figure 3.11: Geodesics between ”0” and ”1”

s



(a) ρ_a to ρ_b at t_1 (b) ρ_a to ρ_b at t_2 (c) ρ_a to ρ_b at t_3 (d) ρ_a to ρ_b at t_4 (e) ρ_a to ρ_b at t_5

(f) ρ_a to ρ_b at t_6 (g) ρ_a to ρ_b at t_7 (h) ρ_a to ρ_b at t_8 (i) ρ_a to ρ_b at t_9 (j) ρ_a to ρ_b at t_{10}

(k) ρ_b to ρ_a at t_1 (l) ρ_b to ρ_a at t_2 (m) ρ_b to ρ_a at t_3 (n) ρ_b to ρ_a at t_4 (o) ρ_b to ρ_a at t_5

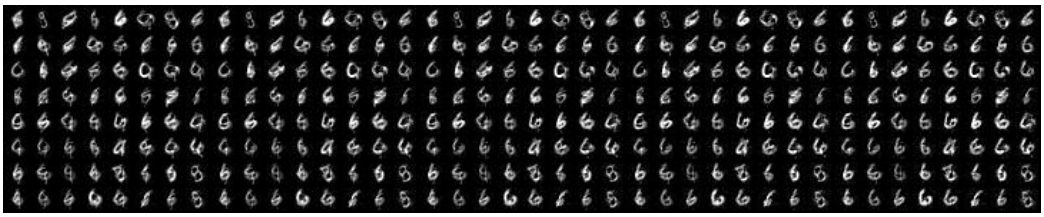
(p) ρ_b to ρ_a at t_6 (q) ρ_b to ρ_a at t_7 (r) ρ_b to ρ_a at t_8 (s) ρ_b to ρ_a at t_9 (t) ρ_b to ρ_a at t_{10}

Figure 3.12: Geodesics between ”4” and ”8”

(a) ρ_a to ρ_b at t_1 (b) ρ_a to ρ_b at t_2 (c) ρ_a to ρ_b at t_3 (d) ρ_a to ρ_b at t_4 (e) ρ_a to ρ_b at t_5

(f) ρ_a to ρ_b at t_6 (g) ρ_a to ρ_b at t_7 (h) ρ_a to ρ_b at t_8 (i) ρ_a to ρ_b at t_9 (j) ρ_a to ρ_b at t_{10}

(k) ρ_b to ρ_a at t_1 (l) ρ_b to ρ_a at t_2 (m) ρ_b to ρ_a at t_3 (n) ρ_b to ρ_a at t_4 (o) ρ_b to ρ_a at t_5



(p) ρ_b to ρ_a at t_6 (q) ρ_b to ρ_a at t_7 (r) ρ_b to ρ_a at t_8 (s) ρ_b to ρ_a at t_9 (t) ρ_b to ρ_a at t_{10}

Figure 3.13: Geodesics between ”6” and ”9”

3.6 Conclusion

OT problem has been drawing more attention in machine learning recently. Though many algorithms have been proposed during the past several years for efficient computations, most of them do not consider the Wasserstein geodesics, neither be suitable for estimating optimal transport map with general cost in high dimensions. In this chapter we present a novel method to compute Wasserstein geodesic between two given distributions with general convex cost. In particular, we consider the primal-dual scheme of the dynamical OT problem and simplify the scheme via the KKT conditions as well as the geodesic transport-

ing properties. By further introducing preconditioning techniques and bidirectional dynamics into our optimization, we obtain a stable and effective algorithm that is capable of high dimensional computation, which is one of the very first scalable algorithms for directly computing geodesics with general cost, including L-p. Our method not only computes sample based Wasserstein geodesics, but also provides Wasserstein distance and optimal map. We demonstrate the effectiveness of our scheme through a series of experiments with both low dimensional and high dimensional settings. It is worth noting that our model can be applied not only in machine learning such as image transfer, density estimation, but also in optimal control and robotics, where one needs to study the distribution of mobile agents. We should also be aware of the malicious usage for our method, for instance, it could be potentially used in some activities involving generating misleading data distributions.

CHAPTER 4

SCALABLE COMPUTATION OF MONGE MAPS WITH GENERAL COSTS

4.1 Introduction

Continuing on discussing OT problem, recall that the dynamic OT 2.4 is beneficial for describing the density evolution under the influence of a vector field [11, 12], but solving it leads to a partial differential equation (PDE) system, which is not a trivial task, especially in a high dimensional setting. To overcome the drawbacks of fluid dynamics formulation, in this chapter we propose a computationally efficient and scalable algorithm for estimating the Wasserstein distance and optimal map between probability distributions over continuous spaces. Particularly, we apply the Lagrangian multiplier on the MK problem, to this end we obtain a minimax problem. Our contribution can be summarized as follows: 1) We develop a scalable algorithm to compute the optimal transport map associated with general transport costs between any two distributions given their samples; 2) Our model is able to deal with the cases that ρ_a or ρ_b is not absolute continuous; 3) Our method is also capable of computing OT problems between distributions over spaces that do not share the same dimension. 4) We provide a rigorous error analysis of the algorithm based on duality gaps; 5) We demonstrate its performance and its scaling properties in truly high dimensional setting through with various experiments, on both synthetic and real data sets.

Related work: [25, 26] propose efficient approaches of computing OT problem, but the methods become infeasible in high dimensional cases. However, Kantorovich dual problem and application of DNN make OT computation more tractable. As a variety of models have been proposed to handle high dimensions via neural networks, we can classify OT models into static ones and dynamic ones. Static models include typical Kantorovich dual based work, such as Wasserstein-1 GAN [7] and Wasserstein-1 related variations [32, 33, 34,

35, 36, 37], where cost function is chosen as L^1 norm, then the Wasserstein-1 distance is computed via solving a maximization problem. However, when we solve for Wasserstein-1 related problems, the non-trivial Lipschitz-1 constraint is challenging to be theoretically and strictly realized. Utilizing input convex neural networks (ICNN) [38] to approximate the potential function is a good way to avoid Lipschitz-1 constraint. For example, [40] and [41] extend the semi-dual formulation of OT to new minimax problems.

The other one track of OT models are dynamic ones. By utilizing the structure of Wasserstein geodesic information [6, 21], several related dual-based formulations are also developed. [56] utilize the geodesic information by defining the Augmented Lagrangian corresponding to Sobolev GAN objective. [23] treat the vector field as a potential function then add optimality condition as a regularizer into the original duality problem. While all of above methods are focusing on either Wasserstein-1 or Wasserstein-2, [57] extends the problem to Wasserstein-p by applying Lagrange multipliers in a bidirectional dynamic scheme, though there is no Lipschitz or convexity constraint in the computation, the algorithm is not very robust and bringing more computation complexity due to the extra time dimension.

The subject of this work originates from the static MK problem. In particular, our algorithm is inspired by the recent work in estimation of optimal transport map and Wasserstein-p distance using dynamic view of OT [57], where samples' density evolution is described via a pushforward map, Lagrange Multipliers are parametrized as neural networks. However our formulation extends to a general cost function setting without considering time dimension.

4.2 Background

Recall the Optimal Transport problem (2.2) on $\mathbb{R}^d$. In this chapter, we consider a more general OT problem from $\mathbb{R}^n$ to $\mathbb{R}^m$

$$\inf_{\pi} \left\{ \int_{\mathbb{R}^n \times \mathbb{R}^m} c(x,y) d\pi(x,y) \;\middle|\; \pi \in \Pi(\rho_a, \rho_b) \right\}. \tag{4.1}$$

We will mainly focus on the cost function $c(\cdot,\cdot)$ satisfying the following conditions:

$$\text{There exists } a \in L^1(\rho_a),\ b \in L^1(\rho_b),\ \text{such that } c(x,y) \geq a(x) + b(y); \tag{4.2}$$

$$c \text{ is } \textbf{locally Lipschitz} \text{ and } \textbf{superdifferentiable} \text{ everywhere;} \tag{4.3}$$

$$\partial_x c(x,\cdot) \text{ is injective for any } x \in \mathbb{R}^n. \tag{4.4}$$

For the definitions of the bold terminologies listed above, please check the Appendix.

The Kantorovich dual problem of the primal OT problem (2.2) is formulated as (Chap 5, [6])

$$\sup_{\substack{(\psi,\phi) \in L^1(\rho_a) \times L^1(\rho_b) \\ \phi(y) - \psi(x) \leq c(x,y)\ \forall\, x \in \mathbb{R}^n, y \in \mathbb{R}^m}} \left\{ \int_{\mathbb{R}^m} \phi(y)\rho_b(y)\, dy - \int_{\mathbb{R}^n} \psi(x)\rho_a(x)\, dx \right\}. \tag{4.5}$$

It is not hard to tell that (4.5) is equivalent to

$$\sup_{\psi \in L^1(\rho_a)} \left\{ \int_{\mathbb{R}^m} \psi^{c,+}(y)\rho_b(y)\, dy - \int_{\mathbb{R}^n} \psi(x)\rho_a(x)\, dx \right\}, \tag{4.6}$$

$$\sup_{\phi \in L^1(\rho_b)} \left\{ \int_{\mathbb{R}^m} \phi(y)\rho_b(y)\, dy - \int_{\mathbb{R}^n} \phi^{c,-}(x)\rho_a(x)\, dx \right\}. \tag{4.7}$$

Here we define $\psi^{c,+}$, $\phi^{c,-}$ via infimum/supremum convolution: $\psi^{c,+}(y) = \inf_x(\psi(x) + c(x,y))$ and $\phi^{c,-}(x) = \sup_y(\phi(y) - c(x,y))$.

The following theorem states the equivalent relationship between the primal Optimal

Transport problem and its Kantorovich dual (4.5). The proof for a more general version can be found in Theorem 5.10 of [6].

Theorem 4.2.1 (Kantorovich Duality)**.** *Suppose c is a cost function defined on $\mathbb{R}^n \times \mathbb{R}^m$ and satisfies* (4.2)*. We denote $C(\rho_a, \rho_b)$ as the infimum value of* (4.1)*. We denote $K(\rho_a, \rho_b)$ as the maximum value of* (4.5)*. Then $C(\rho_a, \rho_b) = K(\rho_a, \rho_b)$.*

In general, the optimal solution π^* to (2.2) can be treated as a random transport plan, i.e. we are allowed to break the single particle into pieces and then transport each piece to certain positions according to the plan π^*. However, in this study, we mainly focus on the classical version of the OT problem, which is known as the **Monge problem** stated as follows

$$\min_{T:\mathbb{R}^n \to \mathbb{R}^m, T_\sharp \rho_a = \rho_b} \int_{\mathbb{R}^n} c(x, T(x)) \rho_a(x)\, dx. \tag{4.8}$$

Here T is a measurable map on $\mathbb{R}^n$, we define the pushforward of distribution ρ_a by T as $T_\sharp \rho_a(E) = \rho_b(T^{-1}(E))$ for any measurable set $E \subset \mathbb{R}^n$. The Monge problem seeks for the optimal deterministic transport plan from ρ_a to ρ_b.

The following result states the existence and uniqueness of the optimal solution to the Monge problem. It is a simplified version of Theorem 10.28 combined with Remark 10.33 taken from [6].

Theorem 4.2.2 (Existence, uniqueness and characterization of the optimal Monge map)**.** *Suppose the cost c satisfies* (4.2)*,* (4.3)*,* (4.4) *, we further assume that ρ_a and ρ_b are compactly supported and ρ_a is absolute continuous with respect to the Lebesgue measure on $\mathbb{R}^n$. Then there exists unique transport map T_* solving the Monge problem* (4.8)*. Furthermore, there exists unique ψ_*, ϕ_* solving the dual problem* (4.5)*, or equivalently, ψ_*, ϕ_* are unique optimal solution of* (4.6)*,* (4.7)*. Then ψ_*, ϕ_* are differentiable on Spt(ρ_a), Spt(ρ_b). And we have*

$$\nabla \psi_*(x) + \partial_x c(x, T_*(x)) = 0,\ \nabla \phi_*(T_*(x)) - \partial_y c(x, T_*(x)) = 0, \quad \rho_a \textit{ almost surely}.$$

4.3 Proposed method

In order to formulate a tractable algorithm for the general Monge problem (4.8), we first notice that (4.8) is a constrained optimization problem. Thus, it is natural to introduce the Lagrange multiplier f for the constraint $T_\sharp \rho_a = \rho_b$ and then reformulate (4.8) as a saddle point problem

$$\sup_f \inf_T \mathcal{L}(T, f) \tag{4.9}$$

with $\mathcal{L}$ defined as

$$\begin{aligned}\mathcal{L}(T, f) &= \int_{\mathbb{R}^n} c(x, T(x))\rho_a(x)dx + \int_{\mathbb{R}^m} f(y)(\rho_b - T_\sharp \rho_a)\, dy \\ &= \int_{\mathbb{R}^n} \left[c(x, T(x)) - f(T(x))\right] \rho_a(x)\, dx + \int_{\mathbb{R}^m} f(y)\rho_b(y)\, dy \end{aligned} \tag{4.10}$$

We can verify that the max-min scheme (4.9) is equivalent to the Kantorovich dual problem (4.7). To this end, one only need to verify:

$$\begin{aligned}\inf_T \mathcal{L}(T, f) &= -\int_{\mathbb{R}^n} \sup_\xi \{f(\xi) - c(x, \xi)\}\rho_a(x)dx + \int_{\mathbb{R}^m} f(y)\rho_b(y)dy \\ &= \int_{\mathbb{R}^m} f(y)\rho_b(y)dy - \int_{\mathbb{R}^n} f^{c,-}(x)\rho_a(x)dx. \end{aligned} \tag{4.11}$$

The following results guarantee that the max-min scheme (4.9) will find the optimal Monge map.

Definition 4.3.1 (Superdifferentiablity)**.** *For function $f : \mathbb{R}^n \to \mathbb{R}$, we say f is superdifferentiable at x, if there exists $p \in \mathbb{R}^n$, such that*

$$f(z) \geq f(x) + \langle p, z - x\rangle + o(|z - x|).$$

Definition 4.3.2 (Locally Lipschitz)**.** *Let $U \subset \mathbb{R}^n$ be open and let $f : \mathbb{R}^n \to \mathbb{R}$ be given.*

Then (1) f is Lipschitz if there exists $L < \infty$ such that

$$\forall x, z \in \mathbb{R}^n, \quad |f(z) - f(x)| \leq L|x - z|.$$

(2) f is said to be locally Lipschitz if for any $x_0 \in \mathbb{R}^n$, there is a neighbourhood O of x_0 in which f is Lipschitz.

Theorem 4.3.1 (Consistency)**.** *Suppose the optimal solution to* (4.9) *is (T_*, f_*), then T_* is the optimal Monge map to the problem* (4.8) *and $f_* = \phi_*$, where ϕ_* is the optimal solution to* (4.7)*.*

Proof. According to (4.11), we are able to tell that the optimal solution f_* equals ϕ_*. Furthermore, at the optimal point (T_*, f_*), we have

$$T_{*\sharp}\rho_a = \rho_b, \quad T_*(x) \in \text{argmax}_{\xi \in \mathbb{R}^m}\{f_*(\xi) - c(x, \xi)\}.$$

The second equation leads to

$$f_*^{c,-}(x) = f_*(T_*(x)) - c(x, T_*(x)).$$

Then we have

$$\begin{aligned}
\int_{\mathbb{R}^n} c(x, T_*(x))\rho_a(x)\, dx &= \int_{\mathbb{R}^n} f_*(T_*(x))\rho_a(x)\, dx - \int_{\mathbb{R}^n} f_*^{c,-}(x)\rho_a(x)\, dx \\
&= \int_{\mathbb{R}^m} f_*(y)\rho_b(y)\, dy - \int_{\mathbb{R}^n} f_*^{c,-}(x)\rho_a(x)\, dx \\
&\leq \int_{\mathbb{R}^n \times \mathbb{R}^m} [f_*(y) - f_*^{c,-}(x)] d\pi(x, y) \leq \int_{\mathbb{R}^n \times \mathbb{R}^m} c(x, y) d\pi(x, y)
\end{aligned}$$

for any $\pi \in \Pi(\rho_a, \rho_b)$. Here the second equality is due to $T_{*\sharp}\rho_a = \rho_b$, the last inequality is due to the definition of $f_*^{c,-}(x) = \sup_y\{f_*(y) - c(x, y)\}$.

As a result, T_* solves (4.8) and thus is the unique optimal Monge map. □

In exact implementation, we will replace both the map T and the dual variable f by the neural networks T_θ, f_η, with θ, η being the parameters of the networks. We aim at solving the following saddle point problem. The algorithm is summarized in Algorithm 2.

$$\max_{\eta} \min_{\theta} \mathcal{L}(T_\theta, f_\eta) := \frac{1}{N} \sum_{k=1}^{N} c(X_k, T_\theta(X_k)) - f_\eta(T_\theta(X_k)) + f_\eta(Y_k) \tag{4.12}$$

where N is the batch size and $\{X_k\}, \{Y_k\}$ are samples generated by ρ_a and ρ_b separately.

Algorithm 2 Computing Wasserstein distance and optimal map from ρ_a to ρ_b

Input: Marginal distributions ρ_a and ρ_b, Batch size N, Cost function $c(x, T(x))$.
Initialize T_θ, f_η.
for K steps **do**
  Sample $\{X_k\}_{k=1}^N$ from ρ_a. Sample $\{Y_k\}_{k=1}^N$ from ρ_b.
  for K_1 steps **do**
    Update (via gradient descent) θ to decrease (4.12)
  end for
  for K_2 steps **do**
    Update (via gradient ascent) η to increase (4.12)
  end for
end for

4.4 Error Analysis via Duality Gaps

In this section, we assume that $m = n = d$, namely, we consider Monge problem between spaces sharing the same dimension d. Suppose that we have solved (4.9) to a certain stage and obtain a pair of (T, f), inspired by the work [58] and [41], we estimate a weighted L^2 error between our computed map T and the optimal Monge map T_*. Firstly let's introduce the definition of c-concave functions. [6].

Definition 4.4.1 (c-concavity)**.** *We say a function $f : \mathbb{R}^d \to \mathbb{R}$ is c-concave if there exists a function φ such that $f = \varphi^{c,+}$. This definition is also equivalent to $(f^{c,-})^{c,+} = f$.*

To accomplish our result, we require ρ_a, ρ_b and $c(x, y)$ to satisfy the conditions men-

tioned in Theorem (4.2.2). Furthermore, we also assume that $c \in C^2(\mathbb{R}^d \times \mathbb{R}^d)$ and satisfies

$$\partial_{xy}c(x,y), \text{ as an } d \times d \text{ matrix, is invertible and self-adjoint.} \tag{4.13}$$

$$\partial_{yy}c(x,y) \text{ is independent of } x; \text{ i.e. } \partial_{yy}c(x,y) = M(y) \text{ with } M(y) \text{ a matrix function of } y. \tag{4.14}$$

We further denote

$$\sigma(x,y) = \sigma_{\min}(\partial_{xy}c(x,y)) \tag{4.15}$$

as the minimum singular value of the matrix $\partial_{xy}c(x,y)$, since it is invertible, $\sigma(x,y) > 0$ for any $x, y \in \mathbb{R}^d$.

Theorem 4.4.1 (Posterior Error Analysis via Duality Gaps)**.** *Assume $f \in C^2(\mathbb{R}^d)$ is a c-concave function and assume that there exists $\varphi \in C^2(\mathbb{R}^d)$ such that $f(y) = \inf_x\{\varphi(x) + c(x,y)\}$.*

Suppose argmin$_x\{\varphi(x) + c(x,y)\}$ is non-empty, finite set for arbitrary $y \in \mathbb{R}^d$. Now for any $\hat{x}_y \in$ argmin$_x\{\varphi(x) + c(x,y)\}$, we further assume there exists function $\lambda(\cdot) > 0$ such that

$$\lambda(y)I_n \succeq \nabla^2_{xx}(\varphi(x) + c(x,y))\Big|_{x=\hat{x}_y} \succ O_n \tag{4.16}$$

We denote the duality gaps

$$\mathcal{E}_1(T,f) = \mathcal{L}(T,f) - \inf_{\widetilde{T}} \mathcal{L}(\widetilde{T},f), \quad \mathcal{E}_2(f) = \sup_{\widetilde{f}} \inf_{\widetilde{T}} \mathcal{L}(\widetilde{T},\widetilde{f}) - \inf_{\widetilde{T}} \mathcal{L}(\widetilde{T},f)$$

Denote T_* as the optimal Monge map of the OT problem (4.8)*. Then there exists a strict positive weight function $\beta(\cdot) > 0$ (depending on c, T_*, f and φ, such that the weighted L^2 error between computed map T and optimal map T_* is upper bounded by*

$$\|T - T_*\|_{L^2(\beta\rho_a)} \leq \sqrt{2(\mathcal{E}_1(T,f) + \mathcal{E}_2(f))}.$$

Before proving the theorem, we need to prove following Lemmas first.

Lemma 4.4.2. *Suppose $n \times n$ matrix A is self-adjoint, i.e. $A = A^T$, with minimum singular value $\sigma_{\min}(A) > 0$. Also assume $n \times n$ matrix H is self-adjoint and satisfies $\lambda I_n \succeq H \succ O_n$. Then $AH^{-1}A \succeq \frac{\sigma_{\min}(A)^2}{\lambda} I_n$.*

Proof. One can first verify that $H^{-1} \succeq \frac{1}{\lambda} I_n$ by digonalizing H^{-1}. To prove this lemma, we only need to verify that for arbitrary $v \in \mathbb{R}^n$,

$$v^{\mathrm{T}} A H^{-1} A v = (Av)^{\mathrm{T}} H^{-1} Av \geq \frac{|Av|^2}{\lambda} \geq \frac{\sigma_{\min}(A)^2}{\lambda} |v|^2$$

Thus $AH^{-1}A - \frac{\sigma_{\min}(A)^2}{\lambda} I_n$ is non-negative definite. □

The following lemma is crucial for proving our results, it analyzes the concavity of the target function $f(\cdot) - c(\cdot, y)$ with f c-concave.

Lemma 4.4.3 (Concavity of $f(\cdot) - c(x, \cdot)$ as f c-concave)**.** *Suppose the cost function $c(x, y)$ and f satisfy the conditions mentioned in Theorem 4.4.1. Denote the function $\Psi_x(y) = f(y) - c(x, y)$, keep all notations defined in Theorem 4.4.1, then we have*

$$\nabla^2 \Psi_x(y) \preceq -\frac{\sigma(x, y)^2}{\lambda(y)} I_n.$$

Proof. First, Notice that f is c-convex, thus, there exists φ such that $f(y) = \inf_x \{\varphi(x) + c(x, y)\}$. Let us also denote $\Phi(x, y) = \varphi(x) + c(x, y)$. Now for a fixed $y \in \mathbb{R}^n$, We pick one

$$\hat{x}_y \in \operatorname{argmin}_x \{\varphi(x) + c(x, y)\}$$

Since we assumed that $\varphi \in C^2(\mathbb{R}^n)$ and $c \in C^2(\mathbb{R}^n \times \mathbb{R}^n)$, we have

$$\partial_x \Phi(\hat{x}_y, y) = \nabla \varphi(\hat{x}_y) + \partial_x c(\hat{x}_y, y) = 0 \tag{4.17}$$

At the same time, since $\hat{x}_y$ is the minimum point of the C^2 function $\Phi(\cdot, y)$, then the Hessian

of $\Phi(\cdot, y)$ at $\hat{x}_y$ is positive definite,

$$\partial_{xx}^2 \Phi(\hat{x}_y, y) = \nabla_{xx}^2 (\varphi(x) + c(x, y))\Big|_{x=\hat{x}_y} = \nabla^2 \varphi(\hat{x}_y) + \partial_{xx}^2 c(\hat{x}_y, y) \succ 0. \tag{4.18}$$

Since $\partial_{xx}^2 \Phi(\hat{x}_y, y)$ is positive definite, it is also invertible. We can now apply the implicit function theorem to show that the equation $\partial_x \Phi(x, y) = 0$ determines an implicit function $\hat{x}(\cdot)$, which satisfies $\hat{x}(y) = \hat{x}_y$ in a small neighbourhood $U \subset \mathbb{R}^n$ containing y. Furthermore, one can show that $\hat{x}(\cdot)$ is continuously differentiable at y. We will denote $\hat{x}_y$ as $\hat{x}(y)$ in our following discussion.

Now differentiating (4.17) with respect to y yields

$$\partial_{xx}^2 \Phi(\hat{x}(y), y) \nabla \hat{x}(y) + \partial_{xy}^2 c(\hat{x}(y), y) = 0. \tag{4.19}$$

On one hand, (4.19) tells us

$$\nabla \hat{x}(y) = -\partial_{xx} \Phi(\hat{x}(y), y)^{-1} \partial_{xy} c(\hat{x}(y), y). \tag{4.20}$$

On the other hand, notice that $c \in C^2(\mathbb{R}^n \times \mathbb{R}^n)$, thus $\partial_{xy} c = \partial_{yx} c$. By (4.19),(4.18), we have

$$\begin{aligned} \partial_{yx}^2 c(\hat{x}(y), y) \nabla \hat{x}(y) =& -\partial_{xx}^2 \Phi(\hat{x}(y), y) \nabla \hat{x}(y) \nabla \hat{x}(y) \\ =& -(\nabla^2 \varphi(\hat{x}(y)) + \partial_{xx}^2 c(\hat{x}(y), y)) \nabla \hat{x}(y) \nabla \hat{x}(y). \end{aligned} \tag{4.21}$$

Now we are able to prove our theorem, we directly compute

$$\nabla^2 \Psi_x(y) = \nabla^2 f(y) - \partial_{yy}^2 c(x, y). \tag{4.22}$$

To calculate $\nabla^2 f(y)$, we first compute $\nabla f(y)$. For fixed y, we suppose $\operatorname{argmin}_x \{\varphi(x) +$

$c(x,y)\} = \{\hat{x}_y^1, \ldots, \hat{x}_y^k\}$. By repeating previous argument, we can obtain k different functions $\{\hat{x}^1(\cdot), \ldots, \hat{x}^k(\cdot)\}$ on a small enough neighbourhood V of y such that $\operatorname{argmin}_x\{\varphi(x)+ c(x,\xi)\} \subset \{\hat{x}^1(\xi), \ldots, \hat{x}^k(\xi)\}$ for any $\xi \in V$. We then denote

$$V_i = \{\xi \mid f(\xi) = \varphi(\hat{x}^i(\xi)) + c(\hat{x}^i(\xi), \xi)\} \cap V.$$

We know $y \in V_i$ for all $1 \leq i \leq k$ and $\bigcup_{i=1}^k V_i = V$. Now if $y \in \text{int}(V_i)$ for certain V_i, i.e., y lies in the interior of V_i, then restricted on V_i, $f(\cdot) = \varphi(\hat{x}^i(\cdot)) + c(\hat{x}^i(\cdot), \cdot)$, we can directly compute the derivative of f as

$$\nabla f(y) = \nabla(\varphi(\hat{x}^i(y)) + c(\hat{x}^i(y), y)) = \partial_y c(\hat{x}^i(y), y),$$

the second equality is due to the envelope theorem [59]. Then $\nabla^2 f(y)$ can be computed as

$$\nabla^2 f(y) = \partial_{yx} c(\hat{x}^i(y), y)\nabla\hat{x}^i(y) + \partial_{yy} c(\hat{x}^i(y), y). \tag{4.23}$$

One the other hand, if y does not belong to the interior of any V_i, we can further prove that there exists V_i such that we can find a small cone with vertex at y belong to V_i. Then, fixed on the cone, we still have $f(\cdot) = \varphi(\hat{x}^i(\cdot)) + c(\hat{x}^i(\cdot), \cdot)$, we can directly compute the first and second order directional derivative of $f(\cdot)$ by computing the derivatives of $\varphi(\hat{x}^i(\cdot)) + c(\hat{x}^i(\cdot), \cdot)$. Finally, since $f \in C^2$, we can still obtain the result in (4.23).

Now for simplicity, we just denote $\hat{x}^i(y)$ as $\hat{x}(y)$, and (4.21) still holds. Plugging (4.21) into (4.23), recall (4.22), this yields

$$\nabla^2 \Psi_x(y) = -(\nabla^2\varphi(\hat{x}(y)) + \partial_{xx}^2 c(\hat{x}(y), y))\nabla\hat{x}(y)\nabla\hat{x}(y) + \partial_{yy}^2 c(\hat{x}(y), y) - \partial_{yy}^2 c(x,y)$$

Now by (4.14), since $\partial_{yy} c(x,y)$ is independent of x, $\partial_{yy}^2 c(\hat{x}(y), y) - \partial_{yy}^2 c(x,y) = 0$. Thus

$\nabla^2\Psi_x(y)$ equals

$$\begin{aligned}\nabla^2\Psi_x(y) &= -(\nabla^2\varphi(\hat{x}(y)) + \partial^2_{xx}c(\hat{x}(y), y))\nabla\hat{x}(y)\nabla\hat{x}(y) \\ &= -\partial^2_{xx}\Phi(\hat{x}(y), y)\nabla\hat{x}(y)\nabla\hat{x}(y) \end{aligned} \tag{4.24}$$

To further simplify (4.24), recall (4.20), we have

$$\nabla^2\Psi_x(y) = -\partial_{xy}c(\hat{x}(y), y)\partial_{xx}\Phi(\hat{x}(y), y)^{-1}\partial_{xy}c(\hat{x}(y), y).$$

By (4.16), we have

$$\lambda(y)I_n \succeq \partial_{xx}\Phi(\hat{x}(y), y) \succ O_n$$

By condition (4.13), $\partial_{xy}c$ is self-adjoint, and (4.15) tells $\sigma_{\min}(\partial_{xy}c(x, y)) = \sigma(x, y)$. Now applying lemma 4.4.2, we have

$$\nabla^2\Psi_x(y) \preceq -\frac{\sigma(x, y)^2}{\lambda(y)}I_n.$$

□

Now we can prove main result in Theorem 4.4.1:

Proof. In this proof, we denote $\int$ as $\int_{\mathbb{R}^d}$ for simplicity. We first recall

$$\mathcal{L}(T, f) = \int f(y)\rho_b(y)\, dy - \int (f(T(x)) - c(x, T(x)))\rho_a(x)dx,$$

also recall definition (4.11), $f^{c,-}(x) = \sup_y\{f(y) - c(x, y)\}$, we can write

$$\begin{aligned}\mathcal{E}_1(T, f) &= -\int [f(T(x)) - c(x, T(x))]\rho_a\, dx + \inf_{\widetilde{T}}\left\{\int [f(\widetilde{T}(x)) - c(x, \widetilde{T}(x))]\rho_a\, dx\right\} \\ &= \int [f^{c,-}(x) - (f(T(x)) - c(x, T(x)))]\rho_a(x)dx\end{aligned}$$

We denote

$$T_f(x) = \text{argmax}_y\{f(y) - c(x,y)\} = \text{argmax}_y\{\Psi_x(y)\},$$

then we have

$$\nabla\Psi_x(T_f(x)) = 0. \tag{4.25}$$

On the other hand, one can write:

$$\begin{aligned}\mathcal{E}_1(T,f) &= \int [(f(T_f(x)) - c(x,T_f(x))) - (f(T(x)) - c(x,T(x)))] \\ &= \int [\Psi_x(T_f(x)) - \Psi_x(T(x))]\rho_a(x)\,dx\end{aligned}$$

For a fixed x, since $\Psi_x(\cdot) \in C^2(\mathbb{R}^n)$, then

$$\Psi_x(T(x)) - \Psi_x(T_f(x)) = \nabla\Psi_x(T_f(x))(T(x) - T_f(x)) + \frac{1}{2}(T(x) - T_f(x))^{\mathrm{T}}\nabla^2\Psi_x(\eta(x))(T(x) - T_f(x))$$

with $\eta(x) = (1-\theta_x)T(x) + \theta_x T_f(x)$ for certain $\theta_x \in (0,1)$. By (4.25) and Lemma 4.4.3, we have

$$\Psi_x(T(x)) - \Psi_x(T_f(x)) \leq -\frac{\sigma(x,\eta(x))^2}{2\lambda(\eta(x))}|T(x) - T_f(x)|^2.$$

Thus we have:

$$\mathcal{E}_1(T,f) = \int [\Psi_x(T_f(x)) - \Psi_x(T(x))]\rho_a(x)\,dx \geq \int \frac{\sigma(x,\eta(x))^2}{2\lambda(\eta(x))}|T(x) - T_f(x)|^2\rho_a(x)\,dx \tag{4.26}$$

On the other hand, let us denote the optimal Monge map from ρ_a to ρ_b as T_*, by Kontorovich duality, we have

$$\sup_f \inf_T \mathcal{L}(T,f) = \inf_{T, T_\sharp\rho_a = \rho_b} \int c(x,T(x))\rho_a\,dx = \int c(x,T_*(x))\rho_a\,dx$$

Thus we have

$$\begin{aligned}\mathcal{E}_2(f) &= \int c(x, T_*(x))\rho_a \, dx - \left(\int f(y)\rho_b \, dy - \int f^{c,-}(x)\rho_a \, dx \right) \\ &= \int c(x, T_*(x))\rho_a \, dx - \left(\int f(T_*(x))\rho_a \, dx - \int f^{c,-}(x)\rho_a \, dx \right) \\ &= \int [f^{c,-}(x) - (f(T_*(x)) - c(x, T_*(x)))]\rho_a \, dx\end{aligned}$$

Similar to the previous treatment, we have

$$\mathcal{E}_2(f) = \int [\Psi_x(T_f(x)) - \Psi_x(T_*(x))]\rho_a(x) \, dx$$

Apply similar analysis as before, we will also have

$$\mathcal{E}_2(f) \geq \int \frac{\sigma(x, \xi(x))^2}{2\lambda(\xi(x))} |T_*(x) - T_f(x)|^2 \rho_a(x) \, dx \tag{4.27}$$

with $\xi(x) = (1 - \tau_x)T_*(x) + \tau_x T_f(x)$ for certain $\tau_x \in (0, 1)$.

Now we set

$$\beta(x) = \min \left\{ \frac{\sigma(x, \eta(x))}{2\lambda(\eta(x))}, \frac{\sigma(x, \xi(x))}{2\lambda(\xi(x))} \right\}, \tag{4.28}$$

combining (4.26) and (4.27), we obtain

$$\begin{aligned}\mathcal{E}_1(T, f) + \mathcal{E}_2(f) &\geq \int \beta(x)(|T(x) - T_f(x)|^2 + |T_*(x) - T_f(x)|^2)\rho_a \, dx \\ &\geq \int \frac{\beta(x)}{2} |T(x) - T_*(x)|^2 \rho_a \, dx\end{aligned}$$

This leads to $\|T - T_*\|_{L^2(\beta\rho_a)} \leq \sqrt{2(\mathcal{E}_1(T, f) + \mathcal{E}_2(f))}$. □

.

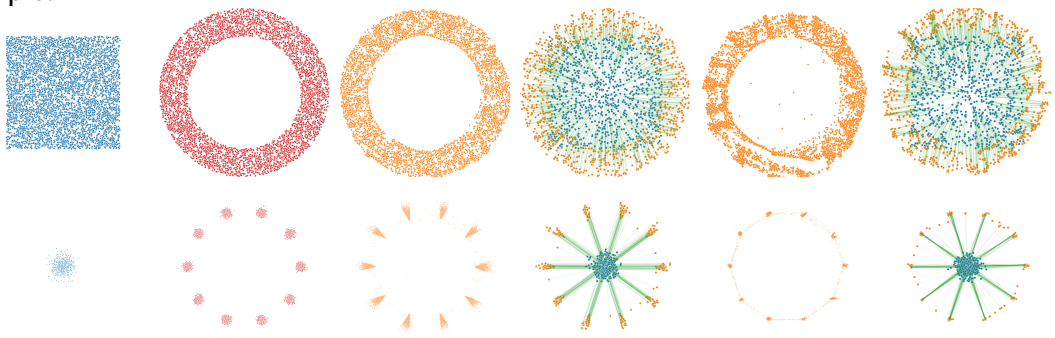
Remark 4. *We can verify that $c(x, y) = \frac{1}{2}|x - y|^2$ or $c(x, y) = -x \cdot y$ satisfy the conditions mentioned above. Then Theorem 4.4.1 recovers similar results proved in [58] and [41].*

Remark 5. *Suppose c satisfies* (4.13) (4.14)*, if c is also an analytical function, then c has the form $\Psi(x) + \nabla u(x)^T y + \Phi(y)$, where Ψ, u, Φ are analytical functions on $\mathbb{R}^d$, and u is strictly convex.*

4.5 Experiments

4.5.1 Learning the 2D optimal map with L^2 cost

In this section, we compare our method with the baseline W2-OT [41]. W2-OT is developed to estimate Wasserstein-2 cost and as such learns the map $\nabla g(\cdot)$ with L^2 cost. As mentioned in related work, it utilizes a min-max optimization structure as well. Figure 4.1 depicts qualitative difference of our algorithm on the Square-Ring and Mixture of ten Gaussian datasets. Each example is represented in a row. It is observed that both methods could learn a reasonable map, however, the samples generated by our method could cover the ρ_b support completely and uniformly. $\nabla g \sharp \rho_a$ leaves some blank space in the Square-Ring example, and excessively concentrates on the ten components in Gaussian mixture example.



(a) ρ_a (b) ρ_b (c) $T\sharp\rho_a$ (d) $T(\cdot)$ map (e) $\nabla g\sharp\rho_a$ (f) $\nabla g(\cdot)$ map

Figure 4.1: Qualitative results of our algorithm for learning 2D map with L^2 cost. The first two columns represent the marginals ρ_a and ρ_b. The marginal distributions are uniformly supported on a square and a ring in the first row. In the second row, they are standard Gaussian and Gaussian mixture with 10 components respectively. The maps generated by our method are demonstrated in (c)-(d) columns and W2-OT maps are in (e)-(f) columns.

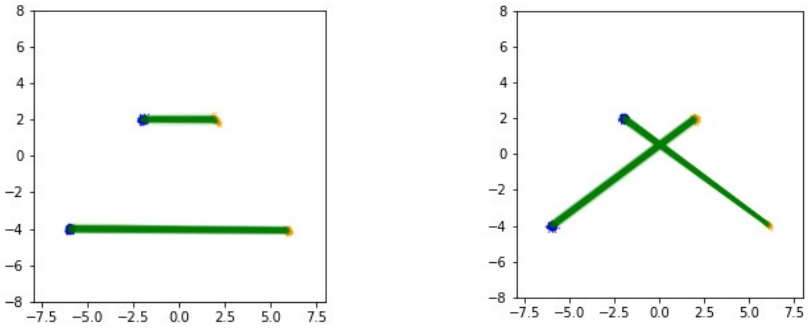


Figure 4.2: Monge map for L^p OT problem from ρ_a to ρ_b, left: $p = 2$, right: $p = 6$

4.5.2 Effect of different costs in 2D space

In Figure 4.2, we consider the Monge problem with the cost $c(x, y) = |x - y|^p$ on $\mathbb{R}^2$, we assume $p > 1$. 0 In order to reflect the difference between L^p-OT problems with different p values, we consider $\rho_a = \frac{1}{2}(\mathcal{N}(\mu_1, \sigma^2 I)+\mathcal{N}(\mu_2, \sigma^2 I))$ and $\rho_b = \frac{1}{2}(\mathcal{N}(\nu_1, \sigma^2 I)+ \mathcal{N}(\nu_2, \sigma^2 I))$, with $\mu_1 = (-2, 2)$, $\mu_2 = (-6, -4)$, $\nu_1 = (2, 2)$, $\nu_2 = (6, -4)$ and $\sigma = 0.1$. Notice that $|\mu_1 - \nu_1|^2 + |\mu_2 - \nu_2|^2 < |\mu_1 - \nu_2|^2 + |\mu_2 - \nu_1|^2$ and $|\mu_1 - \nu_2|^6 + |\mu_2 - \nu_1|^6 < |\mu_1 - \nu_1|^6 + |\mu_2 - \nu_2|^6$, this indicates the difference between the Monge maps of L^2-OT and L^6-OT problems. Such difference is captured by our numerical results shown in Figure 4.2 in the introduction.

4.5.3 Example in 28×28D space

L^2 vs L^1 cost In this experiment we choose MNIST as our data set (28×28 dimensional) and assume each handwritten digit from 0 to 9 follows a specific distribution respectively. Our task is to learn the Monge mapping between 0 and 1, 2 and 5, 3 and 7, 4 and 8, 6 and 9. In each round we treat 0, 2, 3, 4 and 6 as ρ_a and 1, 5, 7, 8, 9 as ρ_b, respectively. We present generated digits that follow ρ_a and ρ_b in Figure 4.3, the learned map T pushforward the digit distributions to target ones. Here we set $\frac{1}{2}||\cdot||_2^2$ as our cost function, thus we are computing Wasserstein-2 distance. With the same training settings, we also show the result of choosing $||\cdot||_1$ as the cost function in Figure 4.4.

(a) $T\sharp\rho_b$ (b) $T\sharp\rho_b$ (c) $T\sharp\rho_b$ (d) $T\sharp\rho_b$ (e) $T\sharp\rho_b$

(f) $T\sharp\rho_a$ (g) $T\sharp\rho_a$ (h) $T\sharp\rho_a$ (i) $T\sharp\rho_a$ (j) $T\sharp\rho_a$

Figure 4.3: Generated ρ_a and ρ_b, $\frac{1}{2}||\cdot||_2^2$ cost

(a) $T\sharp\rho_b$ (b) $T\sharp\rho_b$ (c) $T\sharp\rho_b$ (d) $T\sharp\rho_b$ (e) $T\sharp\rho_b$

(f) $T\sharp\rho_a$ (g) $T\sharp\rho_a$ (h) $T\sharp\rho_a$ (i) $T\sharp\rho_a$ (j) $T\sharp\rho_a$

Figure 4.4: Generated ρ_a and ρ_b, $||\cdot||_1$ cost

4.5.4 Examples in 64×64D space

Recently, several works have illustrated the scalability of our dual formula with different realizations of the transportation costs. With a focus on the quadratic cost, [60] obtains comparable performance in image generative models, which asserts the efficacy in unequal dimension tasks. Similarly, [61] apply the formula with quadratic cost in multiple domain adaptation tasks and achieve the respectable effect. Concurrently, [62] utilizes the dual formula with more diverse costs in image super-resolution task. Our dual formula can be viewed as the extension to all the above approaches.

In this section we show the effectiveness of our method on the inpainting task with

(a) Degraded/original images (b) Composite images $G(x)$ (c) Pushforward images $T(x)$

Figure 4.5: Unpaired image inpainting on **test** dataset of CelebA 64×64. In panel (b) and (c), we show the results with $\alpha = 10$ in the first row and $\alpha = 10000$ in the second row. A small transportation cost would result that pushforward map neglects the connection to the unmasked area, which is illustrated by a clear mask border in pushforward images.

random rectangle masks. We take the distribution of occluded images to be ρ_a and the distribution of the full images to be ρ_b. In many inpainting works, it's assumed that an unlimited amount of paired training data is accessible [63]. However, most real-world applications do not involve the paired datasets. Accordingly, we consider the **unpaired** inpainting task, i.e. no pair of masked image and original image is accessible. The training and test data are generated according to [60, Section 5.2]. We choose cost function to be mean squared error (MSE) in the unmasked area

$$c(x, y) = \alpha \cdot \frac{\|x \odot M - y \odot M\|_2^2}{n},$$

where M is a binary mask with the same size as the image. M takes the value 1 in the unoccluded region, and 0 in the unknown/missing region. $\odot$ represents the point-wise

multiplication, α is a tunable coefficient, and n is dimension of x. Intuitively, this works as a regularization that the pushforward images should be consistent with input images in the unmasked area. Empirically, the map learnt with a larger α can generate more realistic images with natural transition in the mask border and exhibit more details on the face.

We conduct the experiments on CelebA 64×64 and 128×128 datasets [64]. The input images (ρ_a) are occluded by randomly positioned square masks. Each of the source ρ_a and target ρ_b distributions contains $80k$ images. We present the empirical results of inpainting in Figure 4.5 and 4.6. Denote M^C as the complement of M, i.e. $M^C = 1$ in the occluded area and 0 otherwise. We take the composite image $G(x) = T(x) \odot M^C + x \odot M$ as the output image. Additionally, we provide the pushforward images $T(x)$ to illustrate the regularization effect of transportation cost in Figure 4.5.

Figure 4.6: Unpaired image inpainting on **test** dataset of CelebA 128×128.

We also evaluate Fréchet Inception Distance [65] of the generated composite images w.r.t. the original images on the test dataset. We use $40k$ images in total and compute the score with the implementation provided by [66]. The results are presented in Table 4.1. It shows that the transportation cost $c(x, y)$ substantially promotes a map that generates more realistic images.

Table 4.1: Quantitative evaluation results on CelebA 64×64 test dataset.

	$\alpha = 0$	$\alpha = 10$	$\alpha = 10000$
FID	18.7942	9.2857	**3.7109**

4.6 Summary of experiment details

In terms of training, the cost functions we choose, structures of neural networks and details of training process will be introduced here. We ran experiments using a NVIDIA RTX 2080 GPU for the experiments in Section 4.5.1, Section 4.5.3, and NVIDIA RTX 3090 GPU for Section 4.5.4. And we ran experiments on CPU for other experiments. If we train on GPU cards with inner iteration $K_2 = 4$, the training time is about 10 minutes for 2D examples.

For general settings, for all experiments we use the Adam optimizer [55] and vanilla feedforward networks unless specified. The activation functions are all PReLu unless specified.

4.6.1 Normal 2D case

For our method, the networks T_θ and f_η each has 5 layers and 32 hidden neurons. The batch size $N = 100$. $K_1 = 4, K_2 = 1$. The learning rate is 10^{-3}. The number of iterations $K = 12000$.

For the W2-OT, the setup is all the same except $K_1 = 10$. The number of iterations $K = 12000$.

4.6.2 Example in 28×28D space

Here for either L^1 or L^2 case we set both mapping function and the Lagrange multiplier as six layers fully connected neural networks, with ReLU and Tanh activation functions respectively, each layer has 512 nodes, the learning rate is 10^{-4}. The batch size is 500 and we train the model with the same general techniques mentioned before, specially we set K_1 = 8 and K_2 = 1. The algorithm is stable and it converges very fast. For L^1 the average outer iteration to get a convergent result is 8000, total computation time lasts for 3 hours, while for L^2 the average iteration for convergence is 4000 and total computation time is around

1 to 2 hours. From our experiments it seems that L^1 based model is more sensitive to the initial weights of the networks.

4.6.3 Example in 64×64D space

The loss function is slightly different with the (4.10). We modify the $f(T(x))$ to be $f(G(x))$ to strengthen the training of f

$$\sup_{f} \inf_{T} \int_{\mathbb{R}^n} [c(x, T(x)) - f(G(x))] \, \rho_a(x) \, dx + \int_{\mathbb{R}^m} f(y) \rho_b(y) \, dy.$$

In the unpaired inpainting experiments, the images are first cropped at the center with size 140 and then resized to 64×64 or 128×128. We choose learning rate to be $1 \cdot 10^{-3}$, Adam [55] optimizer with default beta parameters, $K_2 = 1$. The batch size is 64 for CelebA64 and 16 for CelebA128. The number of inner loop iteration $K_1 = 5$ for CelebA64 and $K_1 = 10$ for CelebA128.

We use exactly the same UNet for the map T and convolutional neural network for f as [60, Table 9] for CelebA64 and add one additional convolutional block in f network for CelebA128.

On NVIDIA RTX A6000 (48GB), the training time of CelebA64 experiment is 10 hours and the time of CelebA128 is 45 hours.

4.7 Conclusion

In this chapter we present a novel method to compute Monge map between two given distributions with freely chosen cost functions. In particular, we consider applying Lagrange multipliers on MK problem, which leads to a saddle point problem. By further introducing neural networks into our optimization, we obtain a scalable algorithm that can handle most general costs and even the case where the dimensions of marginals are unequal. Our method not only computes sample based Wasserstein distance, but also provides optimal

map. Our scheme is shown to be effective through a series of experiments with both low dimensional and high dimensional settings. It will become an useful tool for machine learning applications such as domain adaption that requires transforming data distributions. It will also be potentially used in areas outside machine learning, such as robotics. On the negative side, since the algorithm is a general tool to transform data distributions, it could be potentially used in many other activities involving maneuvering data distributions.

CHAPTER 5

LEARNING COST FUNCTION FOR OPTIMAL TRANSPORT

5.1 Introduction

In this chapter, we derive an unconstrained convex optimization formulation of the inverse OT problem, which can be further augmented by any customizable regularization. We provide a comprehensive characterization of the properties of inverse OT, including uniqueness of solutions. We also develop two numerical algorithms, one is a fast matrix scaling method based on the Sinkhorn-Knopp algorithm for discrete OT, and the other one is a learning based algorithm that parameterizes the cost function as a deep neural network for continuous OT. The novel framework proposed in the work avoids repeatedly solving a forward OT in each iteration which has been a thorny computational bottleneck for the bi-level optimization in existing inverse OT approaches. Numerical results demonstrate promising efficiency and accuracy advantages of the proposed algorithms over existing state-of-the-art methods.

The discrete version of (2.2) reduces to a linear program. In this case, $\mu \in \Delta^{m-1}$ and $\nu \in \Delta^{n-1}$ become two probability vectors, where $\Delta^{n-1} := \{x = (x_1, \ldots, x_n) \in \mathbb{R}^n : x_i \geq 0,\ \sum_{i=1}^n x_i = 1\}$ stands for the standard probability simplex in $\mathbb{R}^n$, and the cost c and transport plan π each renders an $m \times n$ matrix. Then the discrete OT problem reads

$$\min_{\pi \in \mathbb{R}^{m \times n}} \left\{ \langle c, \pi \rangle \ :\ \pi \geq 0,\ \pi 1_n = \mu,\ \pi^\top 1_m = \nu \right\}, \tag{5.1}$$

where $\langle c, \pi \rangle := \mathrm{tr}(c^\top \pi) = \sum_{i,j} c_{ij} \pi_{ij}$ is the discretized total cost under transport plan π, and $1_n := [1, \ldots, 1]^\top \in \mathbb{R}^n$. We will use these matrix and vector notations exclusively in Sections 5.4.1, 5.5.1, and 5.5.2 during the discussions on discrete inverse OT.

It is important to note that a brute-force discretization of (2.2) yields in general a com-

putationally intractable problem (5.1) when X and Y are in higher dimensional space: m represents the number of grid points (or bins) for discretizing the space $X \subset \mathbb{R}^d$, and hence it grows exponentially fast in d, i.e., $m = N^d$ where N is the discretization points (resolution) in each dimension. Similar for n and Y. This is well known as the "curse of dimensionality" in the literature. Therefore, continuous OT and its inverse problem require a vastly different approach in analysis and computation.

Motivation We consider the inverse problem of OT, i.e., learning the cost function c from observations of the joint distribution π^* or its samples. This work is motivated by a critical issue of OT in all real-world applications: the solution to OT heavily depends on the cost function c, which is the sole latent variable in the OT problem (2.2) to deduce the optimal transport plan for any give pair of marginal distributions (μ, ν). Therefore, the cost function is of paramount importance in shaping the optimal transport plan π^* for further analysis and inference.

In most existing applications of OT, the cost function is simply chosen as a distance-like functions, such as $c(x, y) = \|x - y\|^p$ where $p > 0$, according to users' preferences. However, there are infinitely many choices of p, and cost functions need not be even distance-like in practice. Therefore, a user-chosen cost function may incorrectly estimate the cost incurred to transfer probability masses and thus fail to capture the actual underlying structures and properties of the data. Eventually, a mis-specified cost function results in severely biased optimal transport plan given new marginal distribution pairs, leading to false claims and invalid inferences.

To address the aforementioned issue, we propose to leverage observed pairing data available in practice, which are samples or realizations of the optimal transport plan, to reconstruct the underlying cost function [67, 68, 69, 70, 71]. The reconstructed cost function can be used to study the underlying mechanism of transferring probability/population mass in the application of interests. It can also be used to estimate optimal pairings given new

marginal distributions μ and ν where modeling and computation of OT are involved.

Approach We consider the inverse problem of entropy regularized OT (the reason of using regularized OT over the unregularized counterpart is explained in Section 5.3.2). We propose a variational formulation for learning the cost function such that its induced optimal transport plan is close to the observed transport plan or its samples. This variational formulation yields a bi-level optimization problem, which can be challenging to solve in general. However, by leveraging the dual form of entropy regularized OT, we show that this bi-level optimization can be reformulated as an unconstrained and convex problem in the cost function before adding any customizable regularization.

Based on our new formulation, we develop two efficient numerical algorithms for inverse OT: one for the discrete case and the other for continuous case. In the discrete case, we can realize the observed transport plan $\hat{\pi}$ as a probability matrix, which is either directly given or can be readily summarized using samples. Then we show that the cost c can be computed using a fast matrix scaling algorithm. In the continuous case, $\hat{\pi}$ is often presented by a number of i.i.d. samples in the form of $(x, y) \sim \hat{\pi}$. In this case, we parameterize the cost function as a deep neural network and develop a learning algorithm that is completely mesh-free and thus capable of handling high dimensional continuous inverse OT problems.

A significant advantage of our approach over existing ones is that we can avoid solving a standard OT problem in each iteration in bi-level optimization. To better distinguish from the inverse OT, we hereafter use the term *forward OT* to refer the standard OT problem which solves for the optimal transport plan π^* given cost c. Thus, *the computational complexity our method is comparable to that of a forward OT problem*, which is only a small fraction of complexities of existing bi-level optimization based methods. We will demonstrate this substantial improvement in efficiency in Section 5.5.

Novelty and contributions Existing approaches to the inverse problem of OT aim at recovering the cost function but vary in specific problem formulations and applications

domains. These approaches will be discussed in more details in Section 5.2. Compared to existing ones, our approach is novel and advantageous in the following aspects:

- All existing methods formulate the cost learning as bi-level optimization or its variants, which require *solving the forward OT problem in each outer iteration*. In sharp contrast, our approach yields a convex optimization with customizable regularization, and the complexity of our method is *comparable to the complexity of one forward OT*.

- To the best of our knowledge, the present work is the first in the literature that provides a *comprehensive characterization of the solution(s) of inverse OT*. Moreover, we show that the ill-posedness of inverse OT, particularly the ambiguity issue of unknown cost functions, can be rectified and the ground truth cost function can be recovered robustly under mild conditions.

- Our framework can be applied to *both discrete and continuous settings*. To our best knowledge, the present work is the *first to tackle cost function learning for continuous OT*. This enables the application of inverse OT in a large variety of real-world problems involving high-dimensional data.

Organization The remainder of this chapter is organized as follows. We first provide an overview of existing cost learning approaches in OT and the relations to other metric learning problems in Section 5.2. In Section 5.3, we propose an inverse optimal transport approach for cost function learning, and derive a novel framework based on the dual of the inverse OT formulation. In Section 5.4, we develop two prototype algorithms to recover the cost matrix in the discrete setting and the cost function in continuous setting, and discuss their properties and variations. Numerical experiments and comparisons are provided in Section 5.5. Section 5.10 is the conclusion.

5.2 Related Work

In this section, we add more overviews of OT, inverse OT, and several closely related topics. We also show the relations between cost learning for OT and general metric learning, and contrast our approach to the existing methods.

Computational OT The computation of OT has been a long standing challenge and is still under active research. Most existing work focus on the discrete setting (5.1), which is a special type of linear program (LP). However, the cubic computation complexity for general LP solvers prohibits fast numerical solution for large m and n. In [16], a modification of (5.1) with an additional entropy regularization term in the objective function is proposed:

$$\min_{\pi \in \mathbb{R}^{m \times n}} \left\{ \langle c, \pi \rangle - \varepsilon H(\pi) \,:\, \pi 1_n = \nu,\ \pi^\top 1_m = \nu \right\}, \tag{5.2}$$

where $H(\pi) := -\langle \pi, \log \pi - 1 \rangle = -\sum_{i,j} \pi_{ij} (\log \pi_{ij} - 1)$ is the (normalized) Shannon entropy of π, and $\varepsilon > 0$ is a prescribed weight of the entropy regularization. Due to the entropy term, the troublesome inequality constraint in the original OT (5.1) is eliminated, and the objective function in (5.2) becomes strictly convex which admits unique solution. Moreover, the dual problem of (5.2) is unconstrained, which can be solved by a fast matrix scaling algorithm called the Sinkhorn (or Sinkhorn-Knopp) algorithm [16]. Sinkhorn algorithm has been the common approach to solve (regularized) OT (5.2) numerically in the discrete setting since then. Its property, convergence, and relation to the original OT (5.1) are also extensively studied, for instance, in [72, 73]. A more comprehensive treatment of computational OT in discrete setting, especially in the regularized form, can be found in [74]. The continuous OT problem is considered where the dual variables are parameterized as deep neural networks [75]. The sample complexity of OT is also studied in [76].

Cost Learning for OT The problem of cost learning for optimal transport has received considerable attention in the past few years. In [68, 69], the cost matrix is parameterized

as a bilinear function of the feature vectors of the two sides in optimal transport. The parameter of the bilinear function, i.e., the interaction matrix, is recovered from the observed matchings, which are hypothesized to be samples drawn from the optimal transport plan that maximizes the total social surpluses [68]. The interaction matrix quantifies coupling surplus that are important in the study of quantitative economics. In [70], a primal-dual matrix learning algorithm is proposed to allow more flexible parametrization of the cost matrix and also takes into account inaccurate marginal information for robust learning. In [67], a set of distributions are given where each pair is also associated with a weight coefficient, and the cost matrix is learned by minimizing the weighted sum of EMD between these pairs induced by this cost. Given class labels of documents which are represented as histograms of words, the cost matrix is parameterized as Mahalanobis distance between feature vectors of the words and learned such that the induced EMD between similar documents are small [77]. In [78], the cost matrix is learned such that the induced EMD between histograms labeled as similar are separated from those between dissimilar histograms, which mimics the widely used metric learning setup. In [79], the cost matrix is induced by a kernel mapping, which is jointly learned with a feature-to-label mapping in a label distribution learning framework. This work is extended to a multi-modal, multi-instance, and multi-label learning problem in a follow-up work [80]. In [81], the cost matrix is parameterized as the exponential of negative squared distance between features, where the feature map is learned such that the induced EMD is small for those with same labels and large otherwise. A cost matrix learning method based on the Metropolis-Hasting sampling algorithm is proposed in [71]. In [82], the Sinkhorn iteration is unrolled into a deep neural network with cost matrix as unknown parameter, which is then trained using given side information. The present work targets at the cost learning problem for OT as in the aforementioned ones, but contrasts favorably to them as explained in Section 5.1.

General Metric Learning The aforementioned methods and the work presented in this chapter aim at learning the cost matrix/function, which is related to but different from the standard metric learning [83] in machine learning. In standard metric learning, the goal is to directly learn the distance that quantifies the similarity between features or data points given in the samples. In contrast, the learning problem in inverse OT aims at recovering the cost function (also known as the ground metric) that induces the EMD (or more generally the Wasserstein distance) and optimal transport plan, optimal couplings, or optimal matchings exhibited by the data. In inverse OT, we only observe the couplings/matchings which are not labeled as similar or not. Hence, we cannot directly assess the distance or cost between features. Instead, we need to learn the cost based on the relative frequency of the matchings in the observed data, which is a compounded effect of the cost function and the intra-population competitions. Moreover, the cost function is critical to reveal the underlying mechanism of optimal transport and matchings, and can be used to predict or recommend optimal matchings given new but different marginal distributions [68, 69, 70].

Riemann Distance Learning The cost function learning problem for optimal transport is also related to Riemannian metric learning on probability simplex. In [84, 85], the Riemannian metric, i.e., distances between probability distributions or histograms on the manifold of probablity simplex, is directly learned. In contrast, the goal in this work is to learn the cost function that reveals the interaction between features, which can also automatically induce a metric on the probability simplex if the cost function satisfies proper conditions. Moreover, the learned cost function from inverse OT can be used to provide insights of observed matchings and generate interpretable predictions on new data associations, which are extremely important and useful in many real-world applications.

5.3 Proposed Framework

5.3.1 Preliminaries on Entropy Regularized OT

We consider the entropy regularized OT [16, 74, 86, 87, 72, 88, 89, 90, 91, 92], where the objective function in (2.2) is supplemented by the (negative) entropy of the unknown distribution π. Entropy regularization takes into account of the uncertainty and incompleteness of observed data, which an important advantage over the OT without regularization [67, 68, 69, 74]. The entropy regularized forward OT problem (5.2) is given as follows:

$$\min_{\pi\in\Pi(\mu,\nu)} \left\{ \int_{X\times Y} c(x,y)\, d\pi(x,y) - \varepsilon H(\pi) \right\}, \tag{5.3}$$

where $H(\pi) := -\int (\log(d\pi/d\lambda) + 1)\, d\pi$ denotes the normalized entropy of π (we assume π is absolutely continuous with respect to the Lebesgue measure λ of $X \times Y$ and $d\pi/d\lambda$ denotes the Radon-Nikodym derivative.) Let $\alpha : X \to \mathbb{R}$ and $\beta : Y \to \mathbb{R}$ be the Lagrangian multipliers corresponding to the two marginal constraints in $\Pi(\mu, \nu)$ in (2.2) respectively, we obtain the dual problem of (5.3) as follows,

$$\max_{\alpha,\beta} \left\{ \int_X \alpha\, d\mu + \int_Y \beta\, d\nu - \varepsilon \int_{X\times Y} e^{(\alpha+\beta-c)/\varepsilon}\, d\lambda \right\}. \tag{5.4}$$

Denote (α^c, β^c) the optimal solution of the dual problem (5.4) for the given cost function c, we can readily deduce that the optimal solution π^c to the primal problem (5.3) reads

$$d\pi^c(x,y) = e^{(\alpha^c(x)+\beta^c(y)-c(x,y))/\varepsilon}\, d\lambda, \tag{5.5}$$

which is a closed-form expression of π^c in terms of (α^c, β^c). For notation simplicity, we omit the arguments x and y hereafter when there is no danger of confusion.

5.3.2 Entropy Regularization in Inverse OT

Entropy regularization is particularly important to properly define the inverse problem of OT. To see this, we first consider the discrete OT problem (5.1). Notice that an observation matrix $\hat{\pi}$ containing zero entries does not provide necessary information to fully characterize the cost matrix c: a small $\hat{\pi}_{ij}$ suggests that c_{ij} is relatively large; but if $\hat{\pi}_{ij} = \hat{\pi}_{ik} = 0$ then it is difficult to tell which of c_{ij} and c_{ik} is larger. Although this issue can be somewhat mitigated with additional information on c, it is still a severe problem when $\hat{\pi}$ contains many zeros or such information on c is not available. Indeed, a reasonable inverse OT algorithm needs the relative ratios between the entries (thus better not be zeros) of $\hat{\pi}$, together with the supply distribution given by μ and ν, to infer the underlying cost accurately. However, sparse $\hat{\pi}$ is very common in discrete OT problem (5.1) without any regularization. This is because that an optimal transport plan, as a solution to the linear program (5.1), often occurs at an extremal point of the polytope of the contraint set, and thus the number of nonzero entries is no more than $m + n - 1$ [74]. In this case, the chance to uncover the true cost c is very low. Entropy regularization of OT overcomes this issue as it always yield a transport plan $\hat{\pi}$ with no zero entry. Moreover, as the weight ε approaches 0, the solution of entropy regularized OT tends to that of the standard unregularized OT.

Continuous inverse OT also benefits from entropy regularization. As shown later, our approach is based on the dual formulation of entropy regularized OT. This allows us to parameterize the dual variables and the cost function as deep neural networks and develop an efficient mesh-free method suitable for applications in high-dimensional continuous spaces.

Due to the aforementioned reasons, we consider the inverse problem of entropy regularized OT in the present work. More precisely, we assume that the observation $\hat{\pi}$ was the solution of an entropy regularized OT (5.3) with unknown cost c to be recovered. However, it is important to note that, unlike entropy regularized OT, the inverse problem is *not* sensitive to the weight ε in (5.3). This is because that the solution to (5.3) only depends on c/ε, rather than the actual c. Thus the observation $\hat{\pi}$ only contains information of this

ratio c/ε which is all we can recover. This ratio c/ε provides all information needed no matter which OT one prefers to use for inference and prediction later: the solution to an unregularized OT is invariant to any constant scaling of c; and the solution to an entropy regularized OT can be freely modified by tuning another user-chosen regularization weight $\varepsilon' > 0$ pretending that the given cost is just c/ε.

5.3.3 Inverse OT and Its Dual Formulation

Suppose that the marginal distributions are given as μ and ν, and we observed sample transport plan $\hat{\pi} \in \Pi(\mu, \nu)$ (details about the format of $\hat{\pi}$ will be provided in the next section). Then we propose the following inverse OT problem to learn the underlying cost c from observation $\hat{\pi}$:

$$\min_{c} \quad \mathrm{KL}(\hat{\pi}, \pi^c) + \varepsilon^{-1} R(c), \tag{5.6a}$$

$$\text{s.t.} \quad \pi^c = \underset{\pi \in \Pi(\mu,\nu)}{\arg\min} \left\{ \int_{X \times Y} c \, d\pi - \varepsilon H(\pi) \right\}, \tag{5.6b}$$

where the Kullback-Leibler (KL) divergence between $\hat{\pi}$ and π is defined by

$$\mathrm{KL}(\hat{\pi}, \pi) := \int_{X \times Y} \frac{d\hat{\pi}}{d\pi} \log\left(\frac{d\hat{\pi}}{d\pi}\right) d\pi, \tag{5.7}$$

$d\hat{\pi}/d\pi$ is the Radon-Nikodym derivative of $\hat{\pi}$ with respect to π, $R(c)$ represents the regularization (or constraint) on the cost function c, which is to be specified later, and π^c is the optimal transport plan induced by c, i.e., the solution of (5.3) for any specific c. We multiplied ε^{-1} to $R(c)$ in (5.6a) for notation simplicity later, but $R(c)$ is user-defined and thus can contain ε for cancellation. The model (5.6) is straightforward to interpret: we seek for the cost function c such that the induced π^c is close to the observed transport plan $\hat{\pi}$ in the sense of KL divergence, and meanwhile it respects the specified regularization or satisfies the constraint described by $R(c)$.

The problem (5.6) is a typical bi-level optimization: the upper level problem (5.6a)

involves π^c which is the solution to the minimization in the lower level problem (5.6b). In general, bi-level optimization problems such as (5.6) are considered very challenging to solve: standard bi-level optimization methods require solving the lower level problem (5.6b) during *each update of the variable* c. Therefore, the overall computational cost is very high because the lower level problem, which is an expensive OT problem, needs to be solved for many times (i.e., the number of outer iterations to update c, which can be easily over hundreds or even thousands).

However, we show that the inverse OT problem (5.6) possesses a very special structure. Most notably, we prove that it is equivalent to an unconstrained and convex problem in c before adding $R(c)$ by leveraging the dual form of the entropy regularized OT (5.4). The rigorous statement of this equivalency relation is given in the following theorem.

Theorem 5.3.1. *The bi-level optimization* (5.6) *for inverse OT is equivalent to*

$$\min_{\alpha,\beta,c} \; E(\alpha,\beta,c) + R(c), \tag{5.8}$$

where the functional E *is defined by*

$$E(\alpha,\beta,c) := \int_{X\times Y} c\, d\hat{\pi} - \int_X \alpha\, d\mu - \int_Y \beta\, d\nu + \varepsilon \int_{X\times Y} e^{(\alpha+\beta-c)/\varepsilon}\, d\lambda. \tag{5.9}$$

The equivalency relation is in the sense that c^* *solves* (5.6) *if and only if* $(\alpha^{c^*}, \beta^{c^*}, c^*)$ *solves* (5.8)*, where* $(\alpha^{c^*}, \beta^{c^*})$ *stands for the optimal solution to the dual problem* (5.4) *with cost* c^*.

Proof. Recall that the optimal transport plan π^c induced by c is given in (5.5), where (α^c, β^c) is the optimal solution to the dual problem (5.4). Therefore, (5.5) implies $d\pi^c = \rho\, d\lambda$ where ρ is the density function of π^c given by

$$\rho(x,y) := e^{(\alpha^c(x)+\beta^c(y)-c(x,y))/\varepsilon}. \tag{5.10}$$

Since $\rho > 0$ everywhere, we know $\lambda \ll \pi^c$ and $d\lambda = \rho^{-1} d\pi^c$. On the other hand, since $\hat{\pi} \ll \pi^c \ll \lambda$, we know $d\hat{\pi} = \hat{\rho}\, d\lambda$ for some probability density $\hat{\rho}$. Hence, by the chain rule of measures, we have

$$\begin{aligned} \mathbf{KL}(\hat{\pi}, \pi^c) &= \int_{X\times Y} \frac{\hat{\rho}}{\rho} \log\left(\frac{\hat{\rho}}{\rho}\right) \rho\, d\lambda \\ &= \int_{X\times Y} \hat{\rho} \log \hat{\rho}\, d\lambda - \int_{X\times Y} \log \rho\, d\hat{\pi} \\ &= H(\hat{\pi}) - \varepsilon^{-1} \int_{X\times Y} (\alpha^c + \beta^c - c)\, d\hat{\pi}. \end{aligned} \tag{5.11}$$

Since $\hat{\pi} \in \Pi(\mu, \nu)$, we know

$$\int_{X\times Y} \alpha^c\, d\hat{\pi} = \int_X \alpha^c\, d\mu \quad \text{and} \quad \int_{X\times Y} \beta^c\, d\hat{\pi} = \int_Y \beta^c\, d\nu \tag{5.12}$$

Plugging (5.10) and (5.11) into (5.6a), eliminating the constant $H(\hat{\pi})$ which is independent of c, and multiplying the objective function by $\varepsilon > 0$ which does not alter minimization, we obtain:

$$\min_c \left\{ R(c) - \int_X \alpha^c\, d\mu - \int_Y \beta^c\, d\nu + \int_{X\times Y} c\, d\hat{\pi} \right\}. \tag{5.13}$$

In (5.13), α^c and β^c are the optimal solution of the dual problem of (5.6b) and hence implicitly depend on c. To make them independent variables, we plug (α^c, β^c) into E defined in (5.9) and obtain:

$$\begin{aligned} E(\alpha^c, \beta^c, c) &= \int_{X\times Y} c\, d\hat{\pi} - \int_X \alpha^c\, d\mu - \int_Y \beta^c\, d\nu + \varepsilon \int_{X\times Y} e^{(\alpha+\beta-c)/\varepsilon}\, d\lambda \\ &= \int_{X\times Y} c\, d\hat{\pi} - \int_X \alpha^c\, d\mu - \int_Y \beta^c\, d\nu + \varepsilon, \end{aligned} \tag{5.14}$$

where the last equality is due to the unity property $\int_{X\times Y} d\pi^c = 1$ since $\pi^c \in \Pi(\mu, \nu)$ is a joint probability. On the other hand, for any c, the optimality of (α^c, β^c) to the dual problem

5.4 implies that

$$E(\alpha^c, \beta^c, c) = \min_{\alpha,\beta} E(\alpha, \beta, c). \tag{5.15}$$

Combining (5.14) and (5.15) and plugging to (5.13), merging the minimizations, and eliminating the singled-out constant ε, we obtain (5.8).

To this point, we have showed that, for any fixed c, there is

$$E(\alpha^c, \beta^c, c) + R(c) - \varepsilon = \varepsilon(\mathbf{KL}(\hat{\pi}, \pi^c) + \varepsilon^{-1} R(c)),$$

where the left hand side is the objective function in (5.8) minus the constant ε, and the right hand side is ε multiple of the objective function in (5.6a). Therefore, the two minimization problems are equivalent and share the same set of solutions c^*. □

The variational model (5.8) is the foundation of our algorithmic development for cost function learning in the next section. Compared to (5.6), the optimization problem (5.8) consists of a convex functional E (as shown later) and a customizable regularization (or constraint) R, and hence has the potential to be solved much more efficiently than posed as a bi-level optimization problem (5.6). Notice that, if c is given and fixed, then the inverse OT (5.8) reduces to the dual problem of the (forward) entropy regularized OT (5.4).

As we will show below, the key feature of (5.8) is that the functional $E(\alpha, \beta, c)$ is *jointly convex* in (α, β, c). That is, E is a convex functional defined on

$$\mathcal{W} := \mathcal{C}(X) \times \mathcal{C}(Y) \times \mathcal{C}(X \times Y),$$

where $\mathcal{C}(X)$ stands for the set of all real-valued continuous functions on X. However, unlike (5.4), $E(\alpha, \beta, c)$ is not strictly convex in its variable (α, β, c) and thus we cannot claim uniqueness of its minimizer. Indeed, we will show that there are infinitely many solutions to (5.13) with the same $\hat{\pi}$ when no additional regularization/constraint R is imposed. In order to characterize the solution set of (5.8), we first need to investigate the behavior of E

over the quotient space induced by the following equivalence relation.

Definition 5.3.1. *We say that* (α, β, c) *and* $(\bar{\alpha}, \bar{\beta}, \bar{c})$ *are* equivalent*, denoted by* $(\alpha, \beta, c) \sim (\bar{\alpha}, \bar{\beta}, \bar{c})$*, if* $\alpha(x) + \beta(y) - c(x, y) = \bar{\alpha}(x) + \bar{\beta}(y) - \bar{c}(x, y)$ *for any* $x \in X$ *and* $y \in Y$*.*

It is easy to verify that $\sim$ defines an *equivalence relation* over $\mathcal{W}$ in the classical sense. This equivalence relation induces a *quotient space* $\widetilde{\mathcal{W}} := \mathcal{W}/\sim$. Denote $[(\alpha, \beta, c)] \subset \mathcal{W}$ the equivalence class of (α, β, c). Then $P : \mathcal{W} \to \widetilde{\mathcal{W}}$ defined by $P(\alpha, \beta, c) = [(\alpha, \beta, c)]$ is called the *canonical projection*. We have the following result regarding the functional $E(\alpha, \beta, c)$ in (5.8).

Theorem 5.3.2. *The following statements hold for the functional* $E(\alpha, \beta, c)$ *defined in* (5.8)*:*

(i) $E(\alpha, \beta, c)$ *is jointly convex in* (α, β, c)*;*

(ii) E *is a constant on each equivalence class* $[(\alpha, \beta, c)]$*;*

(iii) Let $\widetilde{E} : \widetilde{\mathcal{W}} \to \mathbb{R}$ *be such that* $\widetilde{E}([(\alpha, \beta, c)]) = E(\alpha, \beta, c)$*, then* $\widetilde{E}$ *is well defined.*

(iv) If (α^*, β^*, c^*) *is a solution of* (5.8)*, then the corresponding optimal transport plan* π^* *in* (5.6) *is given by* $d\pi^* = e^{(\alpha^*(x)+\beta^*(x)-c^*(x,y))/\varepsilon} d\lambda$*. Moreover,* $[(\alpha^*, \beta^*, c^*)]$ *is the unique minimizer of* $\widetilde{E}$*.*

Proof. (i) Let $\phi : X \times Y \to \mathbb{R}^3$ be $\phi(x, y) := (\alpha(x), \beta(y), c(x, y))$ for any $x \in X$ and $y \in Y$. Denote $\zeta = (1, 1, -1) \in \mathbb{R}^3$. Then the functional E in (5.8) is given by

$$E(\phi) = \int_{X\times Y} \phi \cdot (-d\mu, -d\nu, d\hat{\pi}) + \varepsilon \int_{X\times Y} e^{(\zeta\cdot\phi)/\varepsilon}\, d\lambda.$$

For any fixed $\psi : X \times Y \to \mathbb{R}^3$, we define the variation $f : I \to \mathbb{R}$, where $I \subset \mathbb{R}$ is a small open neighborhood of 0, as follows,

$$f(\epsilon) := E(\phi + \epsilon\psi).$$

Then we can verify that

$$f'(\epsilon) = \int_{X\times Y} \psi \cdot (-d\mu, -d\nu, d\hat{\pi}) + \int_{X\times Y} e^{(\zeta\cdot(\phi+\epsilon\psi))/\varepsilon}(\zeta\cdot\psi)\, d\lambda$$

and that

$$f''(\epsilon) = \frac{1}{\varepsilon}\int_{X\times Y} e^{(\zeta\cdot(\phi+\epsilon\psi))/\varepsilon}(\zeta\cdot\psi)^2\, d\lambda \geq 0$$

for any $\epsilon \in I$. Since ψ is arbitrary, we know E is a convex functional of ϕ.

(ii) To show that E is constant over the equivalence class $[\phi] = [(\alpha, \beta, c)]$, we suppose $(\alpha, \beta, c) \sim (\bar{\alpha}, \bar{\beta}, \bar{c})$, i.e., $\alpha(x) + \beta(y) - c(x, y) = \bar{\alpha}(x) + \bar{\beta}(y) - \bar{c}(x, y)$ for all (x, y). Denote $\delta_\alpha(x) := \alpha(x) - \bar{\alpha}(x)$ for every $x \in X$ and $\delta_\beta(y) := \beta(y) - \bar{\beta}(y)$ for every $y \in Y$, then it is obvious that

$$\bar{c}(x, y) = \bar{\alpha}(x) + \bar{\beta}(y) - \alpha(x) - \beta(y) + c(x, y) = -\delta_\alpha(x) - \delta_\beta(y) + c(x, y).$$

Hence we have

$$\begin{aligned}
E(\bar{\alpha}, \bar{\beta}, \bar{c}) &= \int_{X\times Y} \bar{c}\, d\hat{\pi} - \int_X \bar{\alpha}\, d\mu - \int_Y \bar{\beta}\, d\nu + \varepsilon \int_{X\times Y} e^{(\bar{\alpha}+\bar{\beta}-\bar{c})/\varepsilon}\, d\lambda \\
&= \int_{X\times Y} (-\delta_\alpha - \delta_\beta + c)\, d\hat{\pi} - \int_X (\alpha - \delta_\alpha)\, d\mu - \int_Y (\beta - \delta_\beta)\, d\nu + \varepsilon \int_{X\times Y} e^{(\alpha+\beta-c)/\varepsilon}\, d\lambda \\
&= E(\alpha, \beta, c) - \int_{X\times Y} (\delta_\alpha + \delta_\beta)\, d\hat{\pi} + \int_X \delta_\alpha\, d\mu + \int_Y \delta_\beta\, d\nu \\
&= E(\alpha, \beta, c),
\end{aligned}$$

where the last equality is a result of cancellations due to

$$\int_{X\times Y} \delta_\alpha\, d\hat{\pi} = \int_X \delta_\alpha \left(\int_Y d\hat{\pi} \right) = \int_X \delta_\alpha\, d\mu$$
$$\int_{X\times Y} \delta_\beta\, d\hat{\pi} = \int_Y \delta_\beta \left(\int_X d\hat{\pi} \right) = \int_Y \delta_\beta\, d\nu$$

by Fubini theorem. Therefore E is constant throughout the equivalence class $[(\alpha, \beta, c)]$.

(iii) As an immediate consequence of (ii), $\widetilde{E} : \widetilde{\mathcal{W}} \to \mathbb{R}$ with $\widetilde{E}([\phi]) := E(\phi)$ for every $\phi \in \mathcal{W}$ is well defined.

(iv) For any minimizer (α^*, β^*, c^*) of (5.8), we know that (α^*, β^*) minimizes $E(\alpha, \beta, c)$ when $c = c^*$, i.e.,

$$(\alpha^*, \beta^*) = \arg\min_{\alpha,\beta} E(\alpha, \beta, c^*) = \arg\min_{\alpha,\beta} \left\{ \varepsilon \int_{X\times Y} e^{(\alpha+\beta-c^*)/\varepsilon}\, d\lambda - \int_X \alpha\, d\mu - \int_Y \beta\, d\nu \right\}.$$

Hence (α^*, β^*) is the optimal dual variable of the forward OT with cost c^*, and therefore the optimal transport plan (optimal primal variable) is $d\pi^*(x, y) = e^{(\alpha^*(x)+\beta^*(y)-c^*(x,y))/\varepsilon} d\lambda$ for all $(x, y) \in X \times Y$.

To show that $[\phi^*]$ is the unique minimizer of $\widetilde{E}$ over $\widetilde{\mathcal{W}}$ when ϕ^* minimizes E, we define for any nonzero $[\psi] \in \widetilde{\mathcal{W}}$ the variation $g : I \to \mathbb{R}$, where $I \subset \mathbb{R}$ is an open neighborhood of 0, as follows,

$$g(\epsilon) = \widetilde{E}([\phi] + \epsilon[\psi]) = \widetilde{E}([\phi + \epsilon\psi]) = E(\phi + \epsilon\psi),$$

where we used the fact $[\phi] + \epsilon[\psi] = [\phi + \epsilon\psi]$ for any $\phi, \psi \in \mathcal{W}$ and $\epsilon \in I$, which can be easily deduced from Definition 5.3.1, to obtain the second equality. Following the same derivation as in (i), we can show that.

$$g''(\epsilon) = \frac{1}{\varepsilon} \int_{X\times Y} e^{(\zeta\cdot(\phi+\epsilon\psi))/\varepsilon} (\zeta \cdot \psi)^2\, d\lambda.$$

Since $[\psi] \neq 0$, we know $\zeta \cdot \psi(x, y) \neq 0$ at some $(x, y) \in X \times Y$. Since ψ is continuous, we know that there exists $\delta > 0$ and an open neighborhood $U \subset X \times Y$ (with positive measure $\lambda(U) > 0$) of (x, y) such that $e^{(\zeta\cdot(\phi+\epsilon\psi))/\varepsilon}(\zeta \cdot \psi)^2 \geq \delta > 0$ for all $(x, y) \in U$. This implies that

$$g''(\epsilon) \geq \frac{1}{\varepsilon} \int_U \delta\, d\lambda = \varepsilon^{-1}\delta\lambda(U) > 0.$$

Hence g is strictly convex at every $[\phi] \in \widetilde{\mathcal{W}}$. Therefore $[\phi^*]$ is the unique minimizer of $\widetilde{E}$ on the quotient space $\widetilde{\mathcal{W}}$. □

According to Theorem 5.3.2, our inverse OT formulation (5.8) is convex as long as the customizable regularization $R(c)$ is convex in c or imposes a constraint of c onto a convex set. In this case, we can employ convex optimization schemes to solve (5.8) for the cost function, which are computationally much cheaper than solving general bi-level optimizations. Moreover, Theorem 5.3.2 implies that (5.8) admits a unique equivalence set that minimizes the functional E. Therefore, even without $R(c)$, we can characterize the entire optimal solution set using one minimizer of $E(\alpha, \beta, c)$: if (α, β, c) minimizes E, then any other $(\bar{\alpha}, \bar{\beta}, \bar{c})$ minimizes E *if and only if* $(\bar{\alpha}, \bar{\beta}, \bar{c}) \sim (\alpha, \beta, c)$. As we will show later, certain mild regularization $R(c)$ on c can further narrow down the search of the desired cost c to a single point within the equivalence set minimizing E.

5.4 Algorithmic Development

In this section, we develop prototype numerical algorithms for cost function learning based on the inverse OT formulation (5.8). As mentioned in Section 5.1, the cost function reduces to a matrix in the discrete case while renders a real-valued function on $X \times Y$ in the continuous case. Due to this substantial difference, we consider the algorithmic developments in these two cases separately.

5.4.1 Discrete Case

Algorithm for discrete inverse OT In the discrete case, we set the marginal distributions μ and ν are two probability vectors from Δ^{m-1} and Δ^{n-1}, respectively. Thus the cost c and transport plan π are both $m \times n$ matrices. Suppose that we summarize the observed matching pairs into the matching matrix $\hat{\pi}$, e.g., $\pi_{ij} = N_{ij}/N$, where N_{ij} is the number of couples of an individual from the ith class corresponding to μ_i and another individual from the jth class corresponding to ν_j, and $N = \sum_{i=1}^{m} \sum_{j=1}^{n} N_{ij}$, as a result the inverse OT

formulation (5.8) reduces to the following optimization problem:

$$\min_{\alpha,\beta,c} \{R(c) - \langle \alpha, \mu \rangle - \langle \beta, \nu \rangle + \langle c, \hat{\pi} \rangle + s(\alpha, \beta, c)\} \tag{5.16}$$

where $\alpha \in \mathbb{R}^m$, $\beta \in \mathbb{R}^n$, $c \in \mathbb{R}^{m \times n}$, and $s : \mathbb{R}^m \times \mathbb{R}^n \times \mathbb{R}^{m \times n} \to \mathbb{R}$ is defined by

$$s(\alpha, \beta, c) := \varepsilon \sum_{i=1}^{m} \sum_{j=1}^{n} e^{(\alpha_i + \beta_j - c_{ij})/\varepsilon}. \tag{5.17}$$

For the above minimization problem, we apply a variant of matrix scaling by modifying the Sinkhorn-Knopp algorithm that alternately updates α, β, c in (5.16). Specifically, the updates of α and β are identical to that in the Sinkhorn-Knopp algorithm for the forward entropy regularized OT [16]. The update of c reduces to solving a regularized (or constrained, depending on $R(c)$) minimization for fixed α, β. This update scheme is well known as block coordinate descent (BCD) or alternating minimization (AM). In general, convergence of BCD requires joint convexity objective function and Lipschitz continuity of its gradient [93]. The objective function E is shown to be joint convex above, but its gradient is not Lipschitz continuous. We present an alternate formulation which is equivalent to (5.8) but the objective function can be shown to have Lipschitz continuous gradient. We provide details of the BCD algorithm and its convergence for this formulation in last part of this chapter 5.9.

We here advocate a modified algorithm which is easy-to-implement and empirically performs better than BCD in our tests. The updates of α and β remain the same as before. The modification is in the update of c as follows: we split the update of c into two steps, where the first step is matrix scaling to compute an $m \times n$ matrix K such that

$$K_{ij} = \frac{\hat{\pi}_{ij}}{e^{(\alpha_i + \beta_j)/\varepsilon}}, \quad \text{for} \quad i = 1, \ldots, m, \;\; j = 1, \ldots, n,$$

Algorithm 3 Matrix Scaling Algorithm for Cost Learning in Discrete Inverse OT (5.16)

Input: Observed matching matrix $\hat{\pi} \in \mathbb{R}^{m\times n}$ and its marginals $\mu \in \mathbb{R}^m, \nu \in \mathbb{R}^n$.
Initialize: $\alpha \in \mathbb{R}^{m\times 1}, \beta \in \mathbb{R}^{n\times 1}, u = \exp(\alpha/\varepsilon), v = \exp(\beta/\varepsilon), c \in \mathbb{R}^{m\times n}$.
repeat
$K \leftarrow e^{-c/\varepsilon}$
$u \leftarrow \mu/(Kv)$
$v \leftarrow \nu/(K^\top u)$
$K \leftarrow \hat{\pi}/(uv^\top)$
$c \leftarrow \mathrm{prox}_{\gamma R}(-\varepsilon \log(K))$
until convergent
Output: $\alpha = \varepsilon \log u$, $\beta = \varepsilon \log v$, c

and the second step is a proximal gradient descent to obtain the updated c:

$$c = \mathrm{prox}_{\gamma R}(\hat{c}) := \arg\min_{c} \left\{ R(c) + \frac{1}{2\gamma}\|c - \hat{c}\|^2 \right\}, \tag{5.18}$$

where $\hat{c} := -\varepsilon \log K$, and all exponential, logarithm, division operations mentioned here are performed component-wisely. If $R(c)$ imposes a constraint of c to a convex set $C \subset \mathbb{R}^{m\times n}$, then $\mathrm{prox}_{\gamma R}$ reduces to the orthogonal projection onto C, which has unique solution and usually can be computed very fast. We summarize the steps of this modified scheme in Algorithm 3.

It is worth noting that, as discussed in Section 5.3.2, the value of ε is *unidentifiable* in inverse OT given that the data $\hat{\pi}$ only contains information c/ε not c. Hence, we can just set $\varepsilon = 1$ in Algorithm 3 and will recover c/ε, which is the same as c up to a constant scaling.

Uniqueness of solution in discrete inverse OT As inverse problems are underdetermined in general, additional information can be essential to narrow down the search to the desired solution to (5.8). The key is a properly designed $R(c)$ which imposes convex regularization or constraint to convex set. To avoid overloading notations, we use the same symbols to denote the discrete counterparts of those in 5.3.3. For convenience, we denote $J = [1_m^\top \otimes I_n; I_m \otimes 1_n^\top; I_{mn}]$, where I_n is the $n \times n$ identity matrix, and $[\cdot;\cdot]$ stacks the arguments vertically by following the standard MATLAB syntax. Denote

$\phi = [\alpha; \beta; c] \in \mathbb{R}^{m+n+mn}$ (where $c \in \mathbb{R}^{mn}$ stacks the columns of $c \in \mathbb{R}^{m\times n}$ in order vertically, we use the matrix and vector forms of c interchangeably hereafter) , then $J\phi = 0$ iff $\alpha_i + \beta_j = c_{ij}$ for all i, j. The following result characterizes a sufficient condition for unique minimizers of (5.16) if $R(c)$ imposes a convex constraint of c into the set C. Note that, in this case, (5.16) reduces to $\min_{(\alpha,\beta,c)\in\mathcal{W}} E(\alpha, \beta, c)$ over the manifold $\mathcal{W} = \mathbb{R}^m \times \mathbb{R}^n \times C$ describing the constraint on ϕ, and the proximal operator $\mathrm{prox}_{\gamma R}$ in the c-step in Algorithm 3 is the projection onto C.

Theorem 5.4.1. *Suppose $R(c)$ imposes the projection onto a closed convex set C in $\mathbb{R}^{m\times n}$ and* (5.16) *attains minimum at $\phi^* := (\alpha^*, \beta^*, c^*)$. If $T_{\phi^*}\mathcal{W} \cap \ker(J) = \{0\}$, where $T_{\phi^*}\mathcal{W}$ is the tangent space of $\mathcal{W}$ at ϕ^*, then ϕ^* is the unique minimizer of* (5.16)*.*

Proof. In the discrete case, we use the notation $\phi = [\alpha; \beta; c] \in \mathbb{R}^{m+n+mn}$ and $E(\phi) = \langle[-\mu; -\nu; c], \phi\rangle + \varepsilon\langle e^{J\phi/\varepsilon}, 1_{mn}\rangle$, where the exponential is component-wise. Hence the Hessian of $E(\phi)$ is

$$\nabla^2 E(\phi) = \varepsilon^{-1} J^\top \mathrm{diag}(e^{J\phi/\varepsilon}) J \succeq 0.$$

If ϕ^* is a minimizer and $T_{\phi^*}\mathcal{W} \cap \ker(J) = \{0\}$, then for any nonzero $\psi \in T_{\phi^*}\mathcal{W}$, we have $J\psi \neq 0$. Hence,

$$\psi^\top(\nabla^2 E)\psi = \varepsilon^{-1}(J\psi)^\top \mathrm{diag}(e^{J\psi^*/\varepsilon})(J\psi) > 0,$$

since $\mathrm{diag}(e^{J\psi^*/\varepsilon}) \succ 0$. Therefore $\phi^* = (\alpha^*, \beta^*, c^*)$ is the unique global minimizer. □

We can derive closed-form expression for several special cases of C that cover many of those used in practice.

Example 5.4.1 (Symmetric cost matrix)**.** *If C is the set of symmetric matrices with zero diagonal entries, then $\mathrm{prox}_{\gamma R}(\hat{c})$ (γ does not make any difference in this case) in Algorithm*

3, i.e., the projection of $\hat{c}$ onto C, is given by

$$c = \text{prox}_{\gamma R}(\hat{c}) = \frac{\hat{c} + \hat{c}^\top}{2}$$

and followed by setting the diagonal entries c to 0. Despite of its simple projection, the constraint C includes a large number of cost matrices used in practice. In particular, any (multiple of) distance-like cost matrix, i.e., kc with $c_{ij} = |i-j|^p$ for any nonzero $k, p \in \mathbb{R}$ (if $p < 0$ then we require $c_{ii} = 0$ separately) is included in C. Note that any permutation of the indices of such c is still in C.

Importantly, the following corollary shows that the symmetry of c in Example 5.4.1 implies uniqueness of solution to (5.16). Note that we additional assume the diagonal entries of c to be zeros, but this holds in most applications since there is usually no cost incurred when no transfer of probability mass is needed.

Corollary 5.4.1.1. *Suppose $C = \{c \in \mathbb{R}^{n\times n} : c = c^\top, c_{ii} = 0, \forall i\}$ and $R(\cdot)$ is the indicator function of C, i.e., $R(c) = 0$ if $c \in C$ and ∞ otherwise, then* (5.16) *has a unique solution c^*.*

Proof. Since $R(\cdot)$ is the projection onto C, we know $\mathcal{W} = \{\phi = (\alpha, \beta, c) : Lc = 0\}$ where $L \in \mathbb{R}^{(2n^2+n)\times n^2}$ represents the linear mapping such that $Lc \in \mathbb{R}^{2n^2+n}$ stacks vertically the $n \times n$ matrix c, its transpose $c^\top$ and the vector $\text{diag}(c)$ of its diagonal. Therefore $T_\phi \mathcal{W} = \mathcal{W}$ for all $\phi \in \mathcal{W}$.

Suppose $\psi = (\alpha, \beta, c) \in T_{\phi^*}\mathcal{W} \cap \ker(J)$, then $Lc = 0$ and $c_{ij} = \alpha_i + \beta_j$ for all i, j (since $Jc = 0$). Therefore $c_{ii} = \alpha_i + \beta_i = 0$, and hence $\alpha_i = -\beta_i$, for all i. Now we have

$$c_{ij} = \alpha_i + \beta_j = \alpha_i - \alpha_j.$$

Similarly, $c_{ji} = \alpha_j - \alpha_i$. Since c is symmetric, we know that $\alpha_i - \alpha_j = \alpha_j - \alpha_i$, which implies $\alpha_i = \alpha_j$. Therefore $c_{ij} = \alpha_i + \beta_j = \alpha_i - \alpha_j = 0$ for all i, j, and hence $c = 0$.

As a consequence, $\alpha = -\beta = \xi 1_n$ for some constant $\xi \in \mathbb{R}$. So $T_\phi \mathcal{W} \cap \ker(J) = \{(\xi 1_n, -\xi 1_n, 0) \in \mathbb{R}^{2n+n^2} : \xi \in \mathbb{R}\}$.

By Theorem 5.3.2, if $\phi^* = (\alpha^*, \beta^*, c^*)$ solves (5.16), then E attains minimum only at $\{\phi^* + \psi : \psi \in T_\phi \mathcal{W} \cap \ker(J)\} \subset [\phi^*]$, which all contain the same cost matrix c^*. □

Example 5.4.2 (Linear affinity matrix)**.** *The cost matrix c is parameterized as $c = G^\top AD$, where $G = [g_1, \ldots, g_m] \in \mathbb{R}^{p \times m}$ and $D = [d_1, \ldots, d_n] \in \mathbb{R}^{q \times n}$ are given. Here G and D stand for the feature vector matrices for the two populations in matching, and $A \in \mathbb{R}^{p \times q}$ is the so-called* linear affinity matrix *(or interaction matrix) to be reconstructed. Therefore, the constraint set is $C = \{G^\top AD : A \in \mathbb{R}^{p \times q}\}$.*

In the study of personality traits in marriage [68], $g_i \in \mathbb{R}^p$ and $d_j \in \mathbb{R}^q$ are the given feature vectors of the ith class of men and jth class of women in the market for $i \in [m]$ and $j \in [n]$, and $A_{kl} \in \mathbb{R}$ is the complementary coefficient of the kth feature of men and lth feature of women for $k \in [p]$ and $l \in [q]$.

In this case, the projection of $-\varepsilon \log K$ onto the set C in (5.18) *is given by*

$$\operatorname{prox}_R(-\varepsilon (G^+)^\top (\log K) D^+),$$

where G^+ and D^+ are the Moore-Penrose pseudoinverse of G and D, respectively,and can be pre-computed before applying Algorithm 3.

There are many other choices of the constraint set C and the regularization R. These are often application-specific and thus require discussions case by case, which is beyond the scope of the present work. However, the general strategy developed in this section can be easily modified and applied to many different situations.

Other additive regularization. If the regularization $R(c)$ is imposed as an additive penalty term in the objective function in (5.8), then we need to compute the proximal operator $\operatorname{prox}_{\gamma R}$ in the c-step of Algorithm 3 with $\gamma \to 0$ gradually along with iterations. We can

also employ the alternating direction method of multipliers (ADMM) with an additional Lagrange multiplier $\lambda \in \mathbb{R}^{m\times n}$, and execute the c-step as $c \leftarrow \mathrm{prox}_{\gamma R}(-\varepsilon \log K - \lambda)$ followed by the multiplier update step $\lambda \leftarrow \lambda + (c + \varepsilon \log K)$ in Algorithm 3. The main computational cost is still in the proximal operator. For example, if the cost matrix c is known to be low-rank, we can set $R(c) = \|c\|_*$, the nuclear norm of the matrix $c \in \mathbb{R}^{m\times n}$, i.e., the sum of singular values of the matrix c. Nuclear norm is a convex relaxation of the matrix rank and much more computationally tractable than the latter. The closed-form solution of $\mathrm{prox}_{\gamma R}(\tilde{c})$ is $U \max(\Sigma - \gamma I, 0)V^\top$, where $U\Sigma V^\top$ is the singular value decomposition (SVD) of the matrix $\tilde{c}$. Similarly, if the cost matrix c is known to be sparse, we can set $R(c) = \sum_{i=1}^m \sum_{j=1}^n |c_{ij}|$ which is a convex relaxation of the nonzero component counter of c. The proximal operator $c = \mathrm{prox}_{\gamma R}(\tilde{c})$ has a closed form known as the soft shrinkage: $c_{ij} = \mathrm{sign}(\tilde{c}_{ij}) \max(|\tilde{c}_{ij}| - \gamma, 0)$ for all i, j.

5.4.2 Continuous Case

In the continuous case, μ, ν, and π represent probability density functions on X, Y, and $X \times Y$, respectively. Here we are only given i.i.d. samples of these distributions, namely, $x^{(i)} \sim \mu$, $y^{(j)} \sim \nu$, and $(x^{(i)}, y^{(i)}) \sim \hat{\pi} \in \Pi(\mu, \nu)$. We define cost function $c(x, y)$ on the continuous space $X \times Y$ where $X \subset \mathbb{R}^{d_X}$ and $Y \subset \mathbb{R}^{d_Y}$ for potentially high dimensionality d_X and d_Y (e.g., $d_X, d_Y \geq 3$). As mentioned in Section 5.1, discretization of the spaces X and Y renders m and n increasing exponentially fast in d_X and d_Y, and then Algorithm 3 (or any discrete inverse OT algorithm) becomes infeasible computationally.

Cost function parametrization by deep neural networks Our approach (5.8) is an unconstrained optimization with customizable regularization $R(c)$. This allows for a natural solution to overcome the issue of discretization using deep neural networks, which is a significant advantage over bi-level optimization formulations. More precisely, we can parameterize the cost function c, as well as the functions α and β, in (5.8) as deep neural

networks. In this case, α, β, and c are neural networks with output layer dimension 1 and input layer dimensions d_X, d_Y, and $d_X + d_Y$, respectively. In particular, the design of the network architecture of c may take the regularization $R(c)$ into consideration. For example, if $R(c)$ suggests that $c \geq 0$, then we can set the activation function in the output layer as the rectified linear unit (ReLU) $\sigma(x) := \max(x, 0)$. We can also introduce an encoder h_η to be learned, such that the cost c_η is the standard Euclidean distance between encoded features, e.g., $c_\eta(x, y) = |h_\eta(x) - h_\eta(y)|$. Nevertheless, in general, the architectures of the α, β, and c are rather flexible, and can be customized adaptively according to specific applications. We will present several numerical results with architecture specifications used in our experiments.

To formalize our deep neural net approach for solving continuous inverse OT, we let θ denote the parameters of α and β (in actual implementations, α and β are two separate neural networks with different parameters θ_a and θ_b respectively, but we use θ for both to avoid overloaded notations). In addition, we use η to denote the parameters of c. Finally we solve for the optimal pair (θ^*, η^*) by minimizing the loss function L with respect to the network parameters (θ, η) base on (5.8)

$$\min_{\theta,\eta} L(\theta, \eta) := R(c_\eta) - \hat{\mathbb{E}}_\mu[\alpha_\theta] - \hat{\mathbb{E}}_\nu[\beta_\theta] + \hat{\mathbb{E}}_{\hat{\pi}}[c_\eta] + \varepsilon \int_{X\times Y} e^{(\alpha_\theta(x)+\beta_\theta(y)-c_\eta(x,y))/\varepsilon}\, d\lambda. \tag{5.19}$$

In (5.19), the empirical expectations are defined by the sample averages:

$$\hat{\mathbb{E}}_\mu[\alpha_\theta] := \frac{1}{N_\mu}\sum_{i=1}^{N_\mu} \alpha_\theta(x^{(i)}), \quad \hat{\mathbb{E}}_\mu[\beta_\theta] := \frac{1}{N_\nu}\sum_{i=1}^{N_\nu} \beta_\theta(y^{(i)}), \quad \hat{\mathbb{E}}_{\hat{\pi}}[c_\eta] := \frac{1}{N_{\hat{\pi}}}\sum_{i=1}^{N_{\hat{\pi}}} c_\eta(x^{(i)}, y^{(i)}),$$

where $\mathcal{D}_\mu := \{x^{(i)} : i \in [N_\mu]\}$ and $\mathcal{D}_\nu := \{y^{(i)} : i \in [N_\nu]\}$ are i.i.d. samples drawn from μ and ν respectively, and $\mathcal{D}_{\hat{\pi}} := \{(x^{(i)}, y^{(i)}) : i \in [N_{\hat{\pi}}]\}$ are observed pairings of $\hat{\pi}$. If only $\hat{\pi}$ is available, we can also substitute the samples for μ and ν by the first and second coordinates of the samples in $\{(x^{(i)}, y^{(i)}) : i \in [N_{\hat{\pi}}]\}$. The last integral in (5.19) can

be approximated by numerical integration methods, such as Gauss quadrature and sample-based integrations. For example, if $X \times Y$ is bounded, we can sample N_s collocation points $\{(x^{(i)}, y^{(i)}) : i \in [N_s]\}$ from $X \times Y$ uniformly, and approximate the integral by

$$\int_{X \times Y} G_{\theta,\eta}(x, y)\, d\hat{\pi} \approx \frac{1}{N_s} \sum_{i=1}^{N_s} G_{\theta,\eta}(x^{(i)}, y^{(i)}), \tag{5.20}$$

where $G_{\theta,\eta}(x, y) := e^{(\alpha_\theta(x)+\beta_\theta(y)-c_\eta(x,y))/\varepsilon}$. A more appealing method for sample-based integration is to use an importance sampling strategy: we first estimate the mode(s) of the function $G_{\theta,\eta}(x, y)$, and draw i.i.d. samples points $\{(x^{(i)}, y^{(i)}) : i \in [N_s]\}$ from a Gaussian distribution $\rho((x, y); \omega, \Sigma)$ where ω and Σ represent the mean (close to the mode) and variance of the Gaussian (or a mixture of Gaussians), and approximate the integral by

$$\int_{X \times Y} G_{\theta,\eta}(x, y)\, d\hat{\pi} \approx \frac{1}{N_s} \sum_{i=1}^{N_s} \frac{G_{\theta,\eta}(x^{(i)}, y^{(i)})}{\rho((x^{(i)}, y^{(i)}); \omega, \Sigma)}.$$

The advantages of this importance sampling strategy include the capability of integral over unbounded domain $X \times Y$ and smaller sample approximation variance with properly chosen ω and Σ. Other methods for approximating the integrals can also be applied. In our experiments, we simply used the uniform sampling shown in (5.20).

Now we can compute all terms in the loss function $L(\theta, \eta)$ in (5.19). During training process we apply stochastic gradient descent method to L and find an optimal solution (θ^*, η^*). In each iteration, we use all of available samples for the empirical expectations, or we only sample a mini-batch for the computation of the gradient of L with respect to (θ, η). Otherwise, the optimization is standard in deep neural network training. Furthermore, we use scaling $\alpha_\theta \leftarrow \alpha_\theta/\varepsilon$, $\beta_\theta \leftarrow \beta_\theta/\varepsilon$, and $c_\eta \leftarrow c_\eta/\varepsilon$ and hence $L(\theta, \eta)$ can be minimized with $\varepsilon = 1$ in (5.19). Then we scale c_η back by multiplying ε after (θ^*, η^*) is computed. The algorithm is summarized in Algorithm 4.

Algorithm 4 Cost Function Learning for Continuous Inverse OT by minimizing (5.19)

Input: Marginal distributions μ, ν and observed pairing data $\hat{\pi}$.
Initialize: Deep nets $(\alpha_\theta, \beta_\theta)$, c_η.
repeat
1. Draw a mini-batch from $\mathcal{D}_\mu, \mathcal{D}_\nu, \mathcal{D}_{\hat{\pi}}$.
2. Sample $\{(x^{(i)}, y^{(i)}) : i \in [N_s]\} \subset X \times Y$.
3. Form stochastic gradient $\hat{\nabla} L$ with empirical expectations and integral (5.20).
4. Update $(\theta, \eta) \leftarrow (\theta, \eta) - \tau \hat{\nabla} L(\theta, \eta)$.

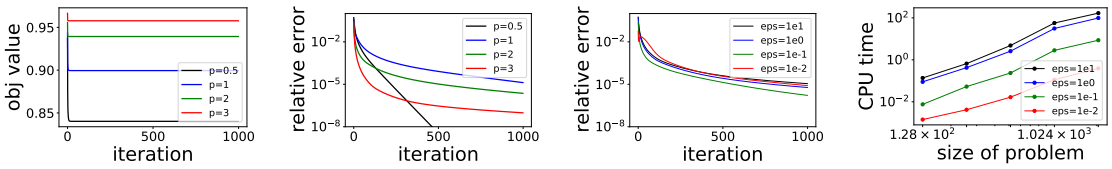
until convergent
Output: α_θ, β_θ, c_η.

5.5 Numerical Experiments

Experiment setup We demonstrate our algorithms (Algorithms 3 and 4) through several synthetic and real data sets. Both algorithms are implemented in Python, where PyTorch is used in Algorithm 4 in the continuous inverse OT problem. The experiments are conducted on a machine equipped with 2.80GHz CPU, 16GB of memory. We use relative error $\|c - c^*\|/\|c^*\|$ to evaluate the cost matrices/functions c learned by the algorithms when the ground truth cost c^* is available. Notice that $\|\cdot\|$ is the standard Frobenius norm of matrices for discrete case. And for continuous case, we evaluate the learned function c_θ and the ground truth cost function c^* at a given finite set of grid points in $X \times Y$, so that both c and c^* can be treated as vectors and the standard 2-norm can be applied.

5.5.1 Discrete Inverse OT on Synthetic Data

We first implement Algorithm 3 on learning cost matrix c using observed transport plan matrix $\hat{\pi}$ in the discrete case. Here we set $m = n$ and ground truth c^* as $c^*_{ij} = |\frac{i-j}{n}|^p$ for $i, j \in [n]$ and $p = 0.5, 1, 2, 3$. Next we use Sinkhorn algorithm [74] to generate $\hat{\pi}$ for each c^* with varying $\varepsilon = 10^1, 10^0, 10^{-1}, 10^{-2}$, and we apply Algorithm 3 to $\hat{\pi}$ and see if we are able to recover the original c^*. To this end, we set the constraint set $C = \{c \in \mathbb{R}^{n \times n} : c = c^\top, c_{ii} = 0, \forall i \in [n]\}$. We also constrain c to be non-negative values by applying $\max(\cdot, 0)$, which seems is able to facilitate this problem.



(a) (b) (c) (d)

Figure 5.1: Results of Algorithm 3 for cost matrix recovery on synthetic data. True $c_{ij} = |\frac{i-j}{n}|^p$ for $i, j \in [n]$. (a) Objective function value versus iteration number for varying p. (b) Relative error (in log scale) versus iteration number for varying p. (c) Relative error (in log scale) versus iteration for varying ε. (d) CPU time (in seconds) versus problem size n in log-log scale for varying ε. Each curve shows the average over 20 instances.

Figure 5.1 shows the results of Algorithm 3. For fixed $\varepsilon = 10^{-1}$, we generate 20 random pairs of $(\mu, \nu) \in \mathbb{R}^m \times \mathbb{R}^n$ and corresponding $\hat{\pi}$ for problem size $m = n = 100$, and apply Algorithm 3 to recover the cost matrix c. Figures 5.1a and 5.1b show the average over the 20 instances of the objective function (5.16) and relative error of c (in logarithm) versus iteration number. In all cases, the true c^* is accurately recovered with relative error approximately 10^{-4} or lower after 500 iterations. For fixed $p = 2$, we also perform the same test of Algorithm 3 on $\hat{\pi}$ generated using varying entropy regularization weight $\varepsilon = 10^1, 10^0, 10^{-1}, 10^{-2}$. We again run 20 instances and plot the relative error (in logarithm) versus iteration. The result is shown in Figure 5.1c. With the same settings for $p = 2$ and varying ε, we test Algorithm 3 for each problem size $n = 128, 256, 512, 1024, 2048$, run the algorithm until the relative error of c reaches 5×10^{-2}, and record the average of the CPU time for 20 instances. We plot the CPU time (in seconds) versus the problem size n (in log-log) in Figure 5.1d. We can see the algorithm run with smaller ε reaches the prescribed relative error in shorter time as Figure 5.1d shows. From Figure 5.1c we see how relative errors decrease as the iteration increases with different ε, all errors converge in similar patterns. These tests evidently show the high efficiency and accuracy of Algorithm 3 in recovering cost matrices in discrete inverse OT.

5.5.2 Discrete Inverse OT on Real Marriage Data

In this experiment we follow the setting of [69] and apply Algorithm 1 to the Dutch Household Survey (DHS) data set (https://www.dhsdata.nl) to estimate the affinity matrix A. Here the cost c has a parametric form $C = -G^\top AD$ where G and D are given feature matrices as described in Section 5.4.1. Following [69], we classify men and women into m and n categories respectively based on the given $G \in \mathbb{R}^{p\times m}$ and $D \in \mathbb{R}^{d\times n}$. These two matrices represent the corresponding feature vectors for men and women. The matrix $A \in \mathbb{R}^{p\times q}$ is the reward (affinity) matrix to be estimated, where the (i, j) entry A_{ij} measures the complementarity or substitutability between the ith attribute of men and the jth attribute of women.

For the historical data used to be trained, we only use the data ranging from 2006 to 2019. We choose 9 features from the data set, namely, educational-level, height, weight, health and 5 personality traits which can be briefly summarized as irresponsible, disciplined, ordered, clumsy, and detail-oriented. All the features are rescaled onto $[0, 1]$ interval. The men and women are both clustered into 5 types by applying k-means algorithm, and each type of men or women is represented by the corresponding cluster center. After data cleaning, the data set contains the information about these features collected from 4,553 couples. In our experiment, we set parameter $\varepsilon = 10^{-2}$. Notice that the initialization of K-means algorithm still affects the values of the estimates, thus we run the experiments 100 times with different fixed random seeds and average the resulting affinity matrix as our final result. We conclude estimated affinity matrix in Table 5.1.

The affinity matrix reveals several important implicit phenomena about marriage market. The education factor gives the most significant complementarity among all the other features. The trade-off between different features is revealed by the off-diagonal coefficients which are significantly different from zero. Since men and women have different preferences for these attributes, the affinity matrix is not symmetric.

We also compare the performance of Algorithm 3 with other pair matching algorithms,

Table 5.1: Affinity matrix estimated using Algorithm 3 on the marriage data. "H" and "W" stand for Husband and Wife respectively, "Edu" stands for Education, "Irres" stands for Irresponsibility, and "Disc" stands for Disciplined.

H\W	Edu	Height	Weight	Health	Irres	Disc	Order	Clumsy	Detail
Edu	0.065	-0.083	-0.052	-0.048	0.015	-0.013	-0.043	-0.063	-0.040
Height	-0.056	-0.461	-0.280	-0.239	-0.054	0.182	-0.232	-0.338	-0.247
Weight	-0.037	-0.301	-0.182	-0.156	-0.037	0.122	-0.151	-0.219	-0.161
Health	-0.035	-0.018	-0.009	-0.014	0.050	-0.125	-0.006	-0.033	-0.009
Irres	-0.017	-0.371	-0.226	-0.188	-0.055	0.215	-0.194	-0.253	-0.202
Disc	-0.002	0.097	0.059	0.055	0.022	0.002	0.050	0.079	0.052
Order	-0.057	-0.309	-0.187	-0.162	-0.034	0.097	-0.150	-0.235	-0.163
Clumsy	-0.020	-0.143	-0.086	-0.075	0.013	0.008	-0.075	-0.107	-0.079
Detail	-0.049	-0.407	-0.247	-0.210	-0.070	0.204	-0.202	-0.295	-0.216

including the state-of-art RIOT model [70], SVD model [94], item-based collaborative filtering model (itemKNN) [95], probabilistic matrix factorization model (PMF) [96], and factorization machine model (FM) [97]. These models have been evaluated on DHS data set in [70] so we just follow the same experimental protocol with 5-fold cross-validation. Note that in [82] the authors come up with a neural network model and achieves the same performance as RIOT on DHS dataset so we just omit the repetitive evaluation here. We train all models on training data set and measure the errors on validation data set by computing the root mean square error (RMSE) and the mean absolute error (MAE). We also run the experiment for 10 times and record the running time for all models. The results are given in Table 5.2. As Table 5.2 shows, our method (Algorithm 3) significantly outperforms all these existing methods in both accuracy and efficiency. Specifically, the RMSE of Algorithm 3 is 2.46×10^{-11} and MAE is 1.90×10^{-11}, so they are rounded as 0.0 in Table 5.2.

5.5.3 Continuous Inverse OT on Synthetic Data

We now apply Algorithm 4 to recover the cost function c in continuous inverse OT. The main difference from the discrete inverse OT is that, instead of learning a cost matrix, we

Table 5.2: Average error of 5-fold cross-validation in RMSE and MAE ($\times 10^{-4}$) and average running time (in seconds) for all compared matching algorithms.

Method	RMSE	MAE	Runtime
Random	45.1	31.4	1.24
PMF [96]	114.5	64.9	0.67
SVD [94]	109.8	62.4	0.73
itemKNN [95]	1.8	1.3	0.97
FM [97]	7.4	5.6	48.51
RIOT [70]	1.8	1.3	7.32
Algorithm 3	**0.0**	**0.0**	**0.04**

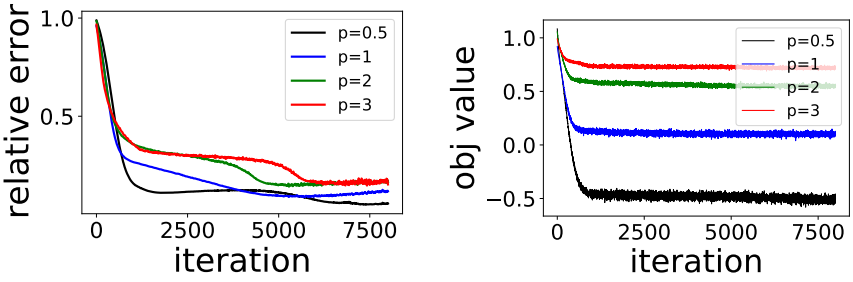
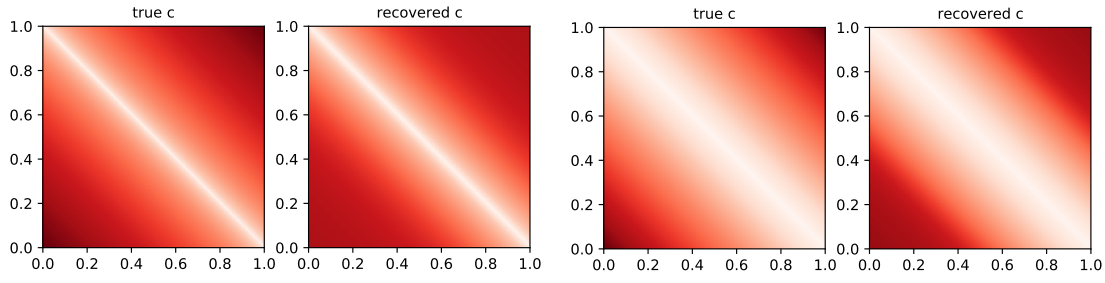


Figure 5.2: The relative error (left) and objective function value (right) versus iteration number by Algorithm 4 on the continuous inverse OT with synthetic data and varying p in the symmetric case.

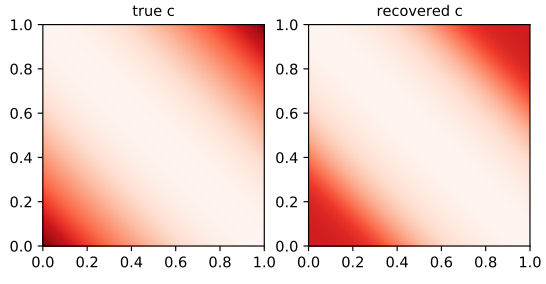
aim at learning a parameterized function $c : X \times Y \to \mathbb{R}$ where $X \subset \mathbb{R}^{d_1}$ and $Y \subset \mathbb{R}^{d_2}$. Here d_1 and d_2 can be 3 or even higher, which causes the issue known as the curse of dimensionality if we discretize X and Y. In this case, we parameterize c as a deep neural network, with input layer size $d_1 + d_2$ and output layer size 1, to overcome the issue of discretization in high-dimensional spaces. For simplicity, we consider the case where $d_1 = d_2$, but the method can be applied to general cases easily.

To justify the accuracy, we first consider the case with $d_1 = d_2 = 1$ so that we can discretize the problem and compute the ground truth optimal transport plan $\hat{\pi}$ accurately for sampling and evaluation purposes. We create a data set $\mathcal{D}_{\hat{\pi}}$ by drawing $N = 5,000$ samples from $\hat{\pi}$ and use them as the sample pairing data for cost learning in each iteration of Algorithm 4. We parameterize c as a 5-layer (including one input layer, 3 hidden layers, and one output layer) deep neural network with 20 neurons per hidden layer, with tanh as

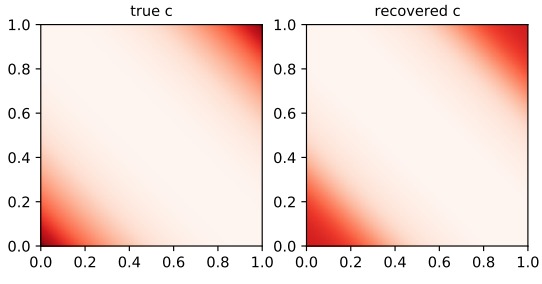


(a) $|x-y|^{1/2}$

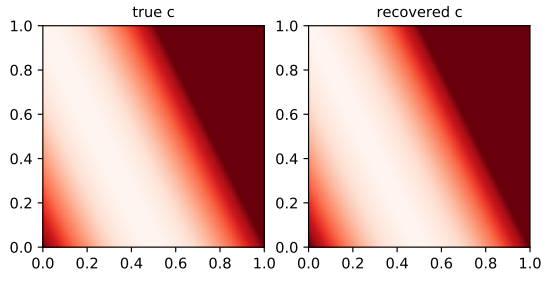
(b) $|x-y|^{1}$



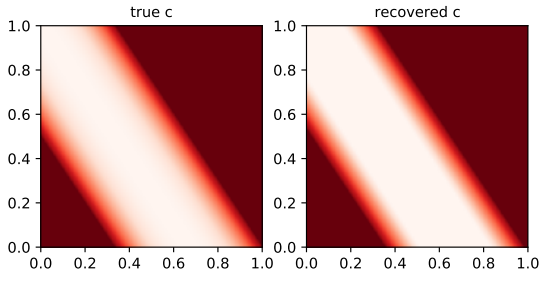
(c) $|x-y|^{2}$



(d) $|x-y|^{3}$



(e) $|x-2y|^{2}$



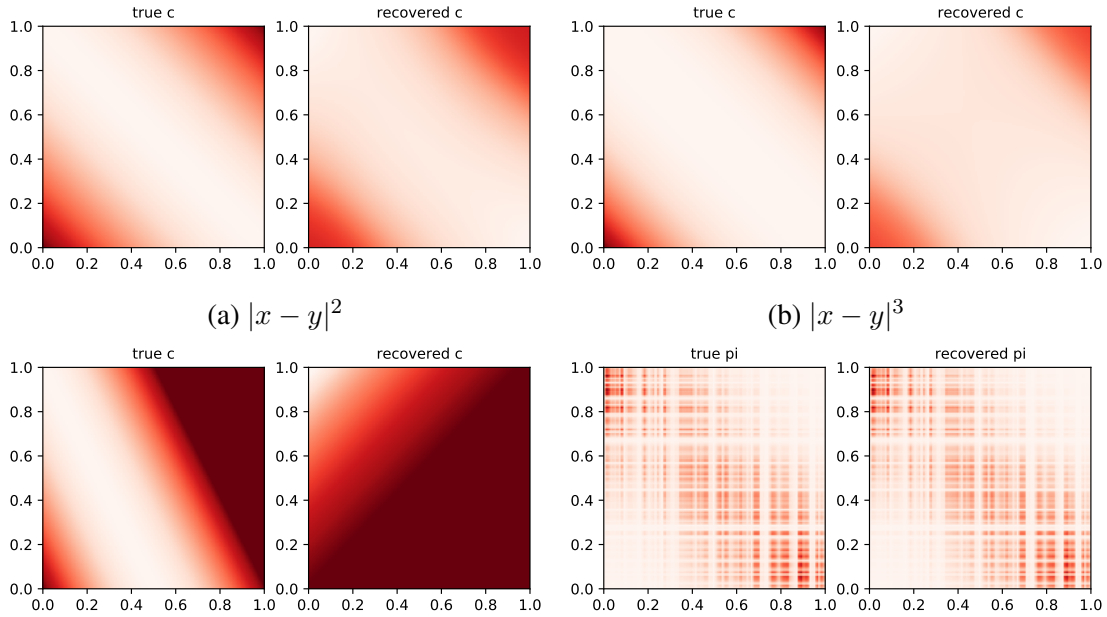
(f) $|2x-3y|^{3}$

Figure 5.3: True cost function and cost function recovered by Algorithm 4 assuming knowledge of the linear proportion between x and y for continuous inverse OT on synthetic data

the activation functions for the hidden layers and ReLU as the output layer.

In the first test, we set the cost to $c = |x-y|^p$ where $p = 0.5, 1, 2, 3$. Here we aim at learning the correct exponent function $(\cdot)^p$ and hence use $|x-y|$ instead of (x, y) as the input (input layer dimension is 1 here). We use PyTorch [98] and the builtin ADAM optimizer [55] with learning rate 10^{-4} for training the network c, where the parameters are initialized using Xavier initialization [99].

In Figure 5.2, we plot the progress of the relative error $\|c - c^*\|_F / \|c^*\|_F$ and objective function value versus iteration number using Algorithm 4. These two plots indicate that both errors of the recovered c and the objective function values obtained Algorithm 4 decay



(a) $|x-y|^2$ (b) $|x-y|^3$

(c) $|x-2y|^2$ (d) $|x-2y|^2$, relative error of π : 0.026

Figure 5.4: True cost function and cost function recovered by Algorithm 4 without knowledge of the proportion between x and y for continuous inverse OT on synthetic data. Notice that (d) shows the optimal transport plan induced by the true cost c (left) and the one by the recovered cost (right) are very similar (with relative error 0.026) despite that the recovered cost differs significantly from the true cost shown in (c). This demonstrates the generic solution non-uniqueness issue of inverse OT if prior knowledge on c is insufficient.

stably. The learned cost functions (image in the right panel) are shown in Figure 5.3 (a)–(d) for $p = 0.5, 1, 2, 3$ respectively, from which we can see that they match the ground truth cost functions (image in the left panel) closely.

We also consider a more challenging problem of recovering asymmetric cost functions $c^*(x,y) = |x-2y|^2$ and $c^*(x,y) = |2x-3y|^3$. We set the input as $\xi = |x-2y|$ and $\xi = |2-3y|$ for the cost function c and again try to recover the unknown exponent $(\cdot)^p$. The network structure and activation functions are set identically to the symmetric case. The ground truth cost and learned cost functions are shown in Figure 5.3e and 5.3f, which demonstrate that Algorithm 4 can also faithfully learn the exponents in the asymmetric case.

Now we conduct a test of Algorithm 4 without any prior information about the cost function. We set the ground truth cost function $c(x,y)$ to be $|x-y|^2$ and $|x-y|^3$, and use

the same generic neural network $c : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ (3 hidden layer, 20 neurons per layer, and tanh and ReLU as the hidden layer activation and output activation respectively). The recovered cost functions are plotted in Figure 5.4a and 5.4b. From 5.4a and 5.4b, we see that Algorithm 4 can still recover the correct cost due to the symmetry.

We again test an asymmetric cost function $c(x, y) = |x - 2y|^2$. We parameterize $c(x, y) = (x - \alpha y)^p$ where both α and p are unknown. Even with such rich prior information about the cost function c, it is difficult to recover the ground truth c faithfully. To see this, we plot the cost function recovered by Algorithm 4 and compare it with the true one in Figure 5.4c. As we can see, the two cost functions are very different. However, when we apply forward OT using these two cost functions, we obtain very similar transport plans with a small relative error 0.026, as shown in Figure 5.4d. This demonstrates the genuine difficulty in the inverse problem of OT: there can be a large number of cost functions that yield the same optimal transport plan as the given one, and it is critically important to impose proper restrictions to c in order to recover the true cost function. Although we proved that this issue can be completely resolved with a mild assumption on the symmetry of c, it still can be a challenging issue in the most general case when such assumption does not hold.

5.5.4 Continuous Inverse OT on Color Transfer

In this test, we consider the inverse OT problem in color transfer between images. Given two RGB color images, the goal of color transfer is to impose the color palette of one image (target) onto the other (source). It is natural to use 3D points to encode the RGB color of pixels, then each image can be viewed as a point cloud in $\mathbb{R}^3$, thus forming a pairing data with $d_X = d_Y = 3$ using optimal transport under certain ground cost c. Specifically, given a cost function c, we can learn the pairing that transfers the point cloud of the source image to the one of the target image by solving a forward OT problem [75]. However, the cost function is critical in shaping the the color transfer result. This experiment is to

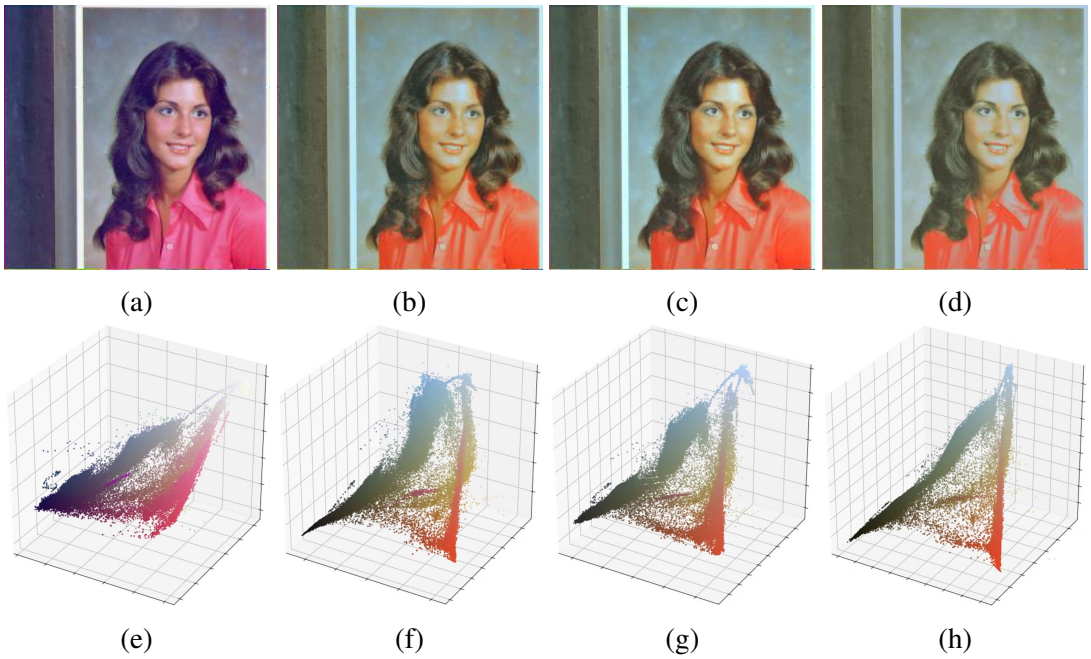


Figure 5.5: Result of color transfer using the cost funciton learned by Algorithm 4. (a) Source image; (b) Target color transferred image; (c) Color transferred image using the cost function learned by Algorithm 4; (d) Color transferred image using mis-specified cost function. Images in the bottom row show the point clouds of color pixels of the images above. The color in image (c) is much more faithful to (b), whereas (d) renders noticeable bias in color fading.

show how an adaptively learned cost using Algorithm 4 can help to overcome the issue with mis-specified cost and avoid inaccurate color transfer.

We obtain a pair of source and target images the USC-SIPI image database Volumn 3 [100]. We set the ground truth cost as $\|x - y\|^2$, and follow [75] to generate the color transfer map. The original and the color transferred images are shown in Figure 5.5. The pairing of point clouds of these two images are used as the samples of $\hat{\pi}$ and fed into Algorithm 4. In Algorithm 4, we parameterize the cost function in the form of $c(x, y) = g((|x_1 - y_1|, |x_2 - y_2|, |x_3 - y_3|))$, where $g : \mathbb{R}^3 \to \mathbb{R}$ is a 5-layer neural network (including 3 hidden layers) with 32 neurons per hidden layer. The activation function is set to tanh. We use ADAM optimizer with learning rate 10^{-3}. After the cost c is learned using Algorithm 4, we test the effect of the learned cost by applying it to the two given point clouds, and show the color transferred image using this learned cost in Figure 5.5c and 5.5g. For comparison,

we also use a mis-specified cost function $c(x, y) = \|x - y\|$ to generate another color transferred image, as shown in Figure 5.5d and its point cloud as 5.5h. As we can see, the image obtained using the cost learned by Algorithm 4 (Figure 5.5c) is much more faithful to the true color (Figure 5.5a), whereas a mis-specified cost function yields an image (Figure 5.5d) with clearly noticeable bias in color tone.

5.6 Remarks on the Implementation of Algorithm 3

The inverse OT problem (5.13) consists of a smooth convex term $E(\alpha, \beta, c)$ defined in (5.9) and a customizable regularization $R(c)$ on c only. The parametrization using deep neural nets introduces non-convexity and hence the convergence issue reduces to that of the SGD (or its variant Adam used in our implementation) in Algorithm 4 for continuous case, and the convergence is still being extensively studied by the community. Thus we here only focus on the convergence for the discrete case. If $R(c)$ imposes a convex regularization or constrains c onto a convex set C, we can directly apply alternating minimization, also known as block coordinate minimization [93], to the blocks α, β, c. In this case, the updates of α and β have closed-form solutions as in Sinkhorn algorithm [16] and also given in Algorithm 3. The update of c, on the other hand, requires solving the following minimization for fixed α, β:

$$\min_c E(\alpha, \beta, c) + R(c) \quad \Longleftrightarrow \quad \min_c \langle c, \hat{\pi} \rangle - \varepsilon \langle e^{(\alpha+\beta-c)/\varepsilon}, 1 \rangle + R(c). \tag{5.21}$$

To handle (5.21), we can either execute one proximal gradient descent step $\mathrm{prox}_{\gamma R}(c - \gamma \nabla_c E(\alpha, \beta, c))$ to update c (hence the problem is not solved exactly), or use multiple iterations to solve it exactly (so inner iterations are needed; other proximal smooth gradient methods can be used as well). The former is equivalent to adding a proximal term of Bregman distance between the latest and next c based on $\nabla_c E$, and the proximal operator is performed only once without solving (5.21) exactly and then we switch to α, β, and so on.

Note that, in either case, the algorithm is *guaranteed to converge* due to the convexity of the problem (5.13). However, despite of the low per-iteration cost of the first approach of handling (5.21), it may require a large number of iterations to converge and hence negatively affect the overall computation complexity. On the other hand, solving (5.21) exactly restrain ourselves in the BCD framework, but we may spend too much time on inner iterations which is not efficient. In what follows, we are more interested in a scheme where a closed-form evaluation is preferred if possible.

For a number of choices of $R(c)$, exact and closed-form solution to (5.21) exists. In particular, if C is given in Corollary 5.4.1.1 which covers a large variety of cost matrices related to norms, the closed-form solution of (5.21) exists and is given by

$$-\varepsilon \log\left(\frac{\hat{\pi}+\hat{\pi}^\top}{uv^\top+vu^\top}\right),$$

where the logarithm and division are performed componentwisely and hence the computation can be done on the fly.

If a closed-form solution of (5.21) is not available, we can turn to the second approach using proximal operator as mentioned above. There is closed-form solution as long as $R(c)$ is *simple*, i.e., its proximal operator prox_R has closed-form solution or is easy to compute. Examples include $\|c\|_*$ (nuclear norm) and $\|c\|_1$ (c interpreted as vector).

In practice, we also find that a modification that splits E and R as presented in Algorithm 3 can further improve convergence speed empirically. This is due to the closed-form of the critical point of $E(\alpha, \beta, c)$ for given α, β. Thus, the per-iteration computational cost remains extremely low in Algorithm 3 as long as prox_R (projection onto C if $R(c)$ imposes constraint to C) is simple to compute, which is true for all the cases mentioned above.

One can also introduce auxiliary variable $\bar{c}$, split $E(\alpha, \beta, c)$ and $R(\bar{c})$ with constraint $\bar{c} = c$, and apply the well-known alternating direction method of multipliers (ADMM) to solve $\alpha, \beta, c, \bar{c}$ with low per-iteration cost, provided that the subproblems are easy to

compute.

In summary, there are a number of choices to numerically solve (5.13) with very high efficiency due to the smooth and convex term E, which is a significant advantage over existing inverse OT approaches based on bi-level optimization. However, the practical implementation for the c step varies significantly according to the choice $R(c)$, which is problem specific and hence not further exploited in this work. Instead, we turn to the general convergence analysis of discrete inverse OT algorithm when the regularization/constraint imposed by $R(c)$ is convex in the next section.

5.7 Characterization of Inverse Problems Bypassing Bi-level Optimization

In general, inverse problems are formulated as optimization problems with complicated constraints. In a typical setting, a set of (unknown) parameters of the problem determines an intermediate variable, and such relation forms the constraint; the intermediate variable is then compared with the measurement data, and the objective (loss) function quantifies their discrepancy. The goal is then to find a (the) set of parameters such that the objective function is minimized.

In the inverse OT problem considered in this work, the cost function c of OT is the set of unknown parameter. The induced optimal transport plan π^c is the intermediate variable, which is compared to the observed data $\hat{\pi}$ using KL divergence as a metric of discrepancy. In neural ODE, the initial value and/or parameters in the defining function (e.g., network parameters) is the set of unknown parameters, the solution of the ODE is the induced intermediate variable, and the objective function is the squared error between the ODE solution and observations. Although the solution of ODE can be considered as an optimal trajectory that fits the ODE determined by the unknown parameters, one often employs numerical ODE solver to approximate the solution, which makes the sense of optimality part a little faded.

When the constraint itself involves an optimization, the constrained problem is called

bi-level optimization, which is considered very challenging computationally. Common approaches to bi-level optimization require solving the optimization problem given in the constrained repeatedly, which results in high computational cost of bi-level optimization.

We showed in this work that the bi-level optimization problem of inverse OT can be reduced to an unconstrained optimization, which is much easier to solve than the original one. Here we provide a general characterizations of problems which may potentially enjoy similar solution approach. In brief, the discrepancy measure in the objective function and the optimization problem described by the constraint need to be paired up to make the reduction possible. To follow the notations used in this work, we use $L(\pi, \gamma; c)$ to denote the Lagrange function of the optimization problem in the constraint. Here π and $\gamma = (\alpha, \beta)$ correspond to the primal and dual variables, and c is the unknown parameter that we want to obtain ultimately. In inverse OT, $L(\pi, \gamma; c) = \langle c, \pi \rangle - \varepsilon H(\pi) - \langle \gamma, A\pi - b \rangle$, where $A\pi = b$ describes the marginal constraint $\pi \in \Pi(\mu, \nu)$ with $A = [1_m^\top \otimes I_n; I_m \otimes 1_n^\top]$ and $b = [\mu; \nu]$. We then define the primal and dual objective functions:

$$F(\pi; c) := L(\pi, \gamma(\pi; c); c) = \max_{\gamma} L(\pi, \gamma; c), \tag{5.22}$$

$$G(\gamma; c) := L(\pi(\gamma; c), \gamma; c) = \min_{\pi} L(\pi, \gamma; c), \tag{5.23}$$

where we used the following notations to denote the optimal dual (primal) variable when the primal (dual) variable is fixed:

$$\gamma(\pi; c) := \arg\max_{\gamma} L(\pi, \gamma; c),$$

$$\pi(\gamma; c) := \arg\min_{\pi} L(\pi, \gamma; c).$$

If the strong duality of L holds (as in OT), we also use $(\pi^*(c), \gamma^*(c))$ to denote the optimal

solution of the minimax problem:

$$\min_{\pi} \max_{\gamma} L(\pi, \gamma; c) = \max_{\gamma} \min_{\pi} L(\pi, \gamma; c) \tag{5.24}$$

Note that $\pi^*(c) = \pi(\gamma^*(c); c) = \arg\min_\pi F(\pi; c)$ and $\gamma^*(c) = \gamma(\pi^*(c); c) = \arg\max_\gamma G(\gamma; c)$.

Consider the bi-level optimization problem of form

$$\min_{c} D(\hat{\pi}, \pi^*(c)), \quad \text{subject to} \quad \pi^*(c) = \arg\min_{\pi} F(\pi; c). \tag{5.25}$$

Suppose that the objective function D is defined by (up to constant shifting and scaling)

$$D(\hat{\pi}, \pi) = F(\hat{\pi}; c) - F(\pi; c) \tag{5.26}$$

for the given data $\hat{\pi}$, then one can readily see that $D(\hat{\pi}, \pi^*(c)) \geq 0$ since $\pi^*(c) = \arg\min_\pi F(\pi; c)$. Moreover $D(\hat{\pi}, \pi^*(c)) = 0$ means that $\hat{\pi}$ is equally good as $\pi^*(c)$ in terms of minimizing $F(\pi; c)$ for the given c. Therefore $D(\hat{\pi}, \pi^*(c))$ is a promising metric to evaluate the quality of a parameter c: the "loss" D is always nonnegative, and vanishes only if $\hat{\pi}$ is a minimizer of $F(\pi; c)$.

In summary, if the objective function in the bi-level optimization problem (5.25) is defined as in (5.26), then the problem can be readily reduced to a simpler unconstrained problem:

$$\begin{aligned}
&\min_{c} \left\{ D(\hat{\pi}, \pi^*(c)) \;\middle|\; \pi^*(c) = \arg\min_{\pi} F(\pi; c) \right\} \\
&= \min_{c} F(\hat{\pi}; c) - F(\pi^*(c); c) \\
&= \min_{c} F(\hat{\pi}; c) - G(\gamma^*(c); c) \\
&= \min_{c} F(\hat{\pi}; c) - \max_{\gamma} G(\gamma; c) \\
&= \min_{c,\gamma} F(\hat{\pi}; c) - G(\gamma; c),
\end{aligned} \tag{5.27}$$

where the second equality is due to the definition of primal and dual objective functions (5.22) and (5.23) and the strong duality (5.24), the third equality is due to the optimality of $\gamma^*(c)$.

The inverse OT formulation we had in this work is exactly one special case of (5.27): for fixed $\varepsilon > 0$, we had $F(\pi; c) = \langle \pi, c \rangle - \varepsilon H(\pi) + \iota_{A\pi=b}(\pi)$ and $G(\gamma; c) = \langle \gamma, b \rangle - \varepsilon \langle e^{(A^\top \gamma - c - \varepsilon)/\varepsilon}, 1 \rangle$, where $\iota_{A\pi=b}(\pi) = 0$ if $A\pi = b$ and $+\infty$ otherwise, and we set the loss function to $D(\hat{\pi}, \pi) = \mathrm{KL}(\hat{\pi} \| \pi)$. Therefore,

$$
\begin{aligned}
D(\hat{\pi}, \pi^*(c)) = \mathrm{KL}(\hat{\pi} \| \pi^*(c)) &= \langle \hat{\pi}, \log(\hat{\pi}/\pi^*(c)) \rangle \\
&= -H(\hat{\pi}) + \langle \hat{\pi}, 1 \rangle - \langle \hat{\pi}, \log \pi(\gamma^*(c); c) \rangle \\
&= -H(\hat{\pi}) + \langle \hat{\pi}, 1 \rangle - \langle \hat{\pi}, \log e^{(A^\top \gamma^*(c) - c)/\varepsilon} \rangle \\
&= -H(\hat{\pi}) + \langle \hat{\pi}, 1 \rangle + \varepsilon^{-1} \langle \hat{\pi}, c \rangle - \varepsilon^{-1} \langle A\hat{\pi}, \gamma^*(c) \rangle \\
&= \varepsilon^{-1} F(\hat{\pi}; c) + \langle \pi^*(c), 1 \rangle - \varepsilon^{-1} \langle A\hat{\pi}, \gamma^*(c) \rangle \\
&= \varepsilon^{-1} F(\hat{\pi}; c) + \langle \pi^*(c), 1 \rangle - \varepsilon^{-1} \langle b, \gamma^*(c) \rangle \\
&= \varepsilon^{-1} F(\hat{\pi}; c) + \langle e^{(A^\top \gamma^*(c) - c)/\varepsilon}, 1 \rangle - \varepsilon^{-1} \langle b, \gamma^*(c) \rangle \\
&= \varepsilon^{-1} F(\hat{\pi}; c) - \varepsilon^{-1} G(\gamma^*(c); c) \\
&= \varepsilon^{-1} (F(\hat{\pi}; c) - F(\pi^*(c); c)),
\end{aligned}
$$

where the third equality is due to $\pi^*(c) = \pi(\gamma^*(c); c)$ as shown under (5.24), the fourth equality is due to the expression of the optimal primal variable using dual variable, the sixth equality is due to the definition of F and that $\hat{\pi}$ and $\pi^*(c)$ are probabilities, the seventh is from $A\hat{\pi} = b$, the ninth due to the definition of G, and the last equality due to strong duality.

5.8 Robust case

Suppose $\hat{\mu}$ and $\hat{\nu}$ are given but noisy, we want to relax the marginal constraint to $\Pi(\mu, \nu)$ where μ and ν are unknown but close to $\hat{\mu}$ and $\hat{\nu}$:

$$\begin{aligned} \min_{c,\mu,\nu} \quad & \mathrm{KL}(\hat{\pi} \,\|\, \pi^c) + R(c), \\ \text{s.t.} \quad & \pi^c = \operatorname*{arg\,min}_{\pi \in \Pi(\mu,\nu)} \langle c, \pi \rangle + \varepsilon \langle \pi, \log \pi - 1 \rangle \\ & D(\mu, \hat{\mu}) \le \rho, \quad D(\nu, \hat{\nu}) \le \rho. \end{aligned}$$

where D is some discrepancy measure, such as squared l_2 or KL. We use relaxation of the last constraint

$$\begin{aligned} \min_{c,\mu,\nu} \quad & \mathrm{KL}(\hat{\pi} \,\|\, \pi^c) + \rho^{-1}(D(\mu, \hat{\mu}) + D(\nu, \hat{\nu})) + R(c), \\ \text{s.t.} \quad & \pi^c = \operatorname*{arg\,min}_{\pi \in \Pi(\mu,\nu)} \langle c, \pi \rangle + \varepsilon \langle \pi, \log \pi - 1 \rangle \end{aligned}$$

Using the same idea we can convert the bi-level problem to an unconstrained problem:

$$\min_{\alpha,\beta,c,\mu,\nu} \; -\langle \alpha, \mu \rangle - \langle \beta, \nu \rangle + \langle c, \hat{\pi} \rangle + \varepsilon \langle e^{(\alpha+\beta-c)/\varepsilon}, 1 \rangle + \rho^{-1}(D(\mu, \hat{\mu}) + D(\nu, \hat{\nu})) + R(c)$$

Note this problem is not jointly convex in all variables. However, alternating minimization still works as it is biconvex. The updates of α, β, c are the same as before.

The update of μ when $D(\mu, \hat{\mu}) = \|\mu - \hat{\mu}\|^2/2$ is:

$$\mu \leftarrow \operatorname*{arg\,min}_{\mu} \frac{1}{2} \|\mu - (\hat{\mu} - \rho\alpha)\|^2 = \mathrm{Proj}_{\Delta^n}(\hat{\mu} - \rho\alpha)$$

which is the projection of $\hat{\mu} - \rho\alpha$ onto the probability simplex Δ^n (closed form solution

exists). The update of μ when $D(\mu, \hat{\mu}) = \mathrm{KL}(\mu||\hat{\mu})$ also has closed-form solution:

$$\mu_i \leftarrow \frac{\hat{\mu}_i e^{-\rho\alpha_i}}{\sum_j \hat{\mu}_j e^{-\rho\alpha_j}}$$

5.9 Block Coordinate Descent for Discrete Inverse OT

In Section 5.4.1, we mentioned that block coordinate descent (BCD) [93] is a widely used approach for solving convex minimization with multiple variables, such as (5.16), when the closed form solution to subproblems are available. However, convergence of BCD requires two key assumptions: the gradient of the objective function is Lipschitz continuous and the iterates generated by BCD are bounded. However, neither of these two assumptions holds for (5.16). To overcome these issues and ensure convergence of BCD, we can reformulate the minimization problem (5.16) into an equivalent form, and show that the convergence can be guaranteed for this new variant of Algorithm 3. This variant of BCD for (5.16) is summarized in Algorithm 5. The reformulation and its equivalency to (5.16) is given in the following lemma.

Lemma 5.9.1. *The inverse OT minimization* (5.16) *is equivalent to the following minimization:*

$$\min_{\alpha,\beta,c} \Psi(\alpha, \beta, c) := F(\alpha, \beta, c) + R(c), \tag{5.28}$$

where the function $E(\alpha, \beta, c)$ *in* (5.16) *is replaced with*

$$F(\alpha, \beta, c) := -\langle \alpha, \mu \rangle - \langle \beta, \nu \rangle + \langle c, \hat{\pi} \rangle + \varepsilon \log\big(\langle e^{(\alpha+\beta-c)/\varepsilon}, 1 \rangle\big).$$

The equivalence is in the sense that (5.16) *and* (5.28) *share exactly the same set of solutions.*

Proof. For any fixed c, we introduce Lagrange multipliers α, β and γ for the equality constraints $\pi 1 = \mu$, $\pi^\top 1 = \nu$, $1^\top \pi 1 = 1$ respectively. Then we can form the dual problem

of the entropy regularized OT:

$$\max_{\alpha,\beta} \langle \alpha, \mu \rangle + \langle \beta, \nu \rangle - \varepsilon \log\big(\langle e^{(\alpha+\beta-c)/\varepsilon}, 1 \rangle\big).$$

The other parts of (5.16) remain the same. Then it is easy to verify that all statements of Theorem 5.3.2 (for discrete setting here) still hold true. Hence (5.16) and (5.28) are equivalent. We omit the details here. □

The main advantages of (5.28) are that the function F is still smooth and convex in (α, β, c), the minimization subproblems (of α and β) still have closed form solutions, and that $\nabla_\alpha F, \nabla_\beta F$, and $\nabla_c F$ are all 1-Lipschitz continuous. The Lipschitiz continuity is a consequence of the following lemma.

Lemma 5.9.2. *For any $a \in \mathbb{R}^n$ and $b \in \mathbb{R}^n_+$, the function $f(x) := \langle a, x \rangle + \log(\sum_{i=1}^n b_i e^{x_i})$ is convex in x, and ∇f is 1-Lipschitz.*

Proof. It is straightforward to verify that $\partial_i f(x) = a_i + \frac{b_i e^{x_i}}{\sum_{j=1}^n b_j e^{x_j}}$. Furthermore, there is

$$\partial^2_{ij} f(x) = \begin{cases} \frac{1}{\|\sqrt{w}\|_2^4}(\|\sqrt{w}\|_2^2 w_i - w_i^2), & \text{if } i = j, \\ -\frac{1}{\|\sqrt{w}\|_2^4}(w_i w_j), & \text{if } i \neq j, \end{cases}$$

where $w_i := b_i e^{x_i}$ for $i = 1, \ldots, n$ and we adopted a slightly misused notation $\sqrt{w} := (\sqrt{w_1}, \ldots, \sqrt{w_n})$. Then for any $\xi \in \mathbb{R}^n$, we can show that

$$\xi^\top \nabla^2 f(x) \xi = \frac{1}{\|\sqrt{w}\|_2^4}(\|\sqrt{w}\|_2^2 \|\sqrt{w}\xi\|_2^2 - |\langle \sqrt{w}, \sqrt{w}\xi \rangle|^2),$$

where $\sqrt{w}\xi := (\sqrt{w_1}\xi_1, \ldots, \sqrt{w_n}\xi_n)$ stands for the componentwise product between $\sqrt{w}$ and ξ. By Cauchy-Schwarz inequality, we have $\langle \sqrt{w}, \sqrt{w}\xi \rangle \leq \|\sqrt{w}\|_2 \cdot \|\sqrt{w}\xi\|_2$, from which it is clear that $\xi^\top \nabla f(x) \xi \geq 0$. Hence f is convex in x. Furthermore, there is

$\|\sqrt{w}\xi\|_2^2 \leq \|\sqrt{w}\|_2^2 \cdot \|\xi\|_2^2$, from which we can see that

$$\xi^\top \nabla^2 f(x)\xi \leq \frac{\|\sqrt{w}\xi\|_2^2}{\|\sqrt{w}\|_2^2} \leq \|\xi\|_2^2,$$

which implies that ∇f is 1-Lipschitz. □

To apply BCD with guaranteed convergence, we also need the boundedness of the iterates $x = (\alpha_k, \beta_k, c_k)$. In the literature of BCD or alternating minimization (AM), an assumption on the boundedness of the sub-level set $\{x : \Psi(x) \leq \Psi(x_0)\}$ or that Ψ is coercive is needed. However, neither of these holds for the (5.28). To ensure boundedness of the iterates, we can restrict our search of the cost matrix c such that $0 \leq c_{ij} \leq M_c$ for some $M_c > 0$ in addition to the constraint or regularization enforced by $R(c)$. However, we do not have similar bounded restrictions on α and β. To overcome this issue, we need to shift the solution of each minimization subproblem of BCD for (5.28) without affecting its optimality. To this end, we need the following definition.

Definition 5.9.1. *A set $S \subset \mathbb{R}^n$ is said to have* bounded variation $M \in [0, \infty)$ *if*

$$\sup_{x \in S} \max_{1 \leq i,j \leq n} |x_i - x_j| \leq M.$$

Note that the requirement of bounded variation of S is weaker than the boundedness of S. Now we can show that the solution sets of the minimization subproblems in α and β both have bounded variations in the following lemma. Note that we can always eliminate the zero components of μ and ν and regard them as strictly positive probability vectors.

Lemma 5.9.3. *For any c and any β, the set $\arg\min_\alpha F(\alpha, \beta, c)$ has bounded variation $M_\alpha := M_c + \varepsilon \log(\mu_{\max}/\mu_{\min})$. Similarly, for any α, the set $\arg\min_\alpha F(\alpha, \beta, c)$ has bounded variation $M_\beta := M_c + \varepsilon \log(\nu_{\max}/\nu_{\min})$. Here $\mu_{\max}$ and $\mu_{\min}$ stand for the largest and smallest components of μ respectively.*

Algorithm 5 Block Coordinate Descent (BCD) for Discrete Inverse OT (5.16)

Input: Observed matching matrix $\hat{\pi} \in \mathbb{R}^{m\times n}$ and its marginals $\mu \in \mathbb{R}^m, \nu \in \mathbb{R}^n$.
Initialize: $\alpha \in \mathbb{R}^{m\times 1}, \beta \in \mathbb{R}^{n\times 1}, u = \exp(\alpha), v = \exp(\beta)$, $c \in \mathbb{R}^{m\times n}$, $c_{ij} \in [0, M_c]$.
repeat
$K \leftarrow e^{-c}$
$u \leftarrow \mu/(Kv)$ and rescale u by κ such that $e^{-M_\alpha} \le \lambda u \le e^{M_\alpha}$
$v \leftarrow \nu/(K^\top u)$ and rescale v by κ such that $e^{-M_\beta} \le \kappa v \le e^{M_\beta}$
$c \in \arg\min_{0\le c_{ij}\le M_c} R(c) + F(\log u, \log v, c)$
until convergent
Output: $\alpha = \varepsilon \log u$, $\beta = \varepsilon \log v$, $c = \varepsilon c$.

Proof. For any c and β, we can check the optimality condition of an arbitrary $\alpha^* \in \arg\min_\alpha F(\alpha, \beta, c)$. This condition is given by $\nabla_\alpha F(\alpha^*, \beta, c) = 0$, which yields

$$\frac{e^{\alpha_i^*/\varepsilon} \sum_j e^{(\beta_j - c_{ij})/\varepsilon}}{\sum_{i,j} e^{(\alpha_i^* + \beta_j - c_{ij})/\varepsilon}} = \mu_i. \tag{5.29}$$

Taking logarithm of both sides and recalling the notation s in (5.17), we obtain

$$\frac{\alpha_i^*}{\varepsilon} = \log \mu_i + \log \varepsilon^{-1} s(\alpha^*, \beta, c) - \log\left(\sum_j e^{(\beta_j - c_{ij})/\varepsilon}\right).$$

Since $0 \le c_{ij} \le M_c$, we know $e^{-M_c/\varepsilon} \le e^{-c_{ij}/\varepsilon} \le 1$, and hence from the equality above we obtain

$$\varepsilon \log\left(\frac{e^{\beta_j/\varepsilon} s(\alpha^*, \beta, c)}{\varepsilon}\right) + \varepsilon \log \mu_i \le \alpha_i^* \le M_c + \varepsilon \log\left(\frac{e^{\beta_j/\varepsilon} s(\alpha^*, \beta, c)}{\varepsilon}\right) + \varepsilon \log \mu_i.$$

Therefore the variation of α^*, i.e., $\max_{1\le i,j\le n} |\alpha_i^* - \alpha_j^*|$, is bounded by $M_c + \varepsilon \log(\mu_{\max}/\mu_{\min})$. The proof for β^* is similar and hence omitted. □

Now we are ready to establish the convergence of Algorithm 5. For simplicity, we directly apply rescaling of $\alpha \leftarrow \alpha/\varepsilon, \beta \leftarrow \beta/\varepsilon, c \leftarrow c/\varepsilon$ which results in an equivalent problem of (5.28) before Algorithm 5 starts, and rescale them back once the computation is finished. Due to Lemma 5.9.3, we can always perform a shifting $\alpha \leftarrow \alpha - t1$ such that

$\|\alpha\|_\infty \leq M_c/2$. The shifting constant $t \in \mathbb{R}$ can be simply set to $(\alpha_{(1)} - \alpha_{(n)})/2$, where $\alpha_{(1)}$ and $\alpha_{(n)}$ stand for the largest and smallest components of α, respectively. As we can see, such shifting does not alter the optimality of α and it still satisfies (5.29). Also note that this shifting is equivalent to rescaling $u = e^\alpha$ into $[e^{-M_c}, e^{M_c}]$ by $\kappa = e^t$, as presented in Algorithm 5. The convergence of Algorithm 5 is given in the following theorem.

Theorem 5.9.4. *Let (α_k, β_k, c_k) be the sequence generated by the BCD Algorithm 5 from any initial (α_0, β_0, c_0), then*

$$0 \leq \Psi_k - \Psi^* \leq \min\left\{\frac{2}{9D^2} - 2, 2, \Psi_0 - \Psi^*\right\}\frac{18D^2}{k}, \tag{5.30}$$

where $\Psi_k := \Psi(\alpha_k, \beta_k, c_k)$ and $D^2 = mM_\alpha^2 + nM_\beta^2 + mnM_c^2$.

Proof. By Lemma 5.9.2 we know $\nabla_\beta F(\alpha, \beta, c)$, $\nabla_\alpha F(\alpha, \beta, c)$ and $\nabla_c F(\alpha, \beta, c)$ are 1-Lipschitz continuous. Moreover, Algorithm 5 is equivalent to the standard BCD with where the iterates lie in the bounded set $\{(\alpha, \beta, c) : \|\alpha\|_\infty \leq M_\alpha,\ \|\beta\|_\infty \leq M_\beta,\ \|c\|_\infty \leq M_c\}$ due to Lemma 5.9.3. By invoking [101, Theorem 2(a)], we obtain (5.30). □

5.10 Conclusion

In this chapter, we conduct a comprehensive study of the inverse problem for OT, i.e., learning the cost function given transport plan observations. We propose a novel inverse OT approach to learn the cost functions such that the induced OT plan is close to the observed plan or its samples. Unlike the bi-level optimization in the literature, we derive a novel formulation to learn the cost function by minimizing an unconstrained convex functional, which can be further augmented by customizable regularization on the cost. We provide a comprehensive characterization of the inverse problem, including the structure of its solution and mild conditions that yield solution uniqueness. We also developed two prototype numerical algorithms to recover the cost in the discrete and continuous settings separately. Numerical results show very promising efficiency and accuracy of our approach.

CHAPTER 6

LEARNING STOCHASTIC BEHAVIOUR FROM AGGREGATE DATA

6.1 Introduction

In the context of dynamical systems, **Aggregate data** refers to a data format in which the full trajectory of each individual modeled by the evolution of state is not available, but rather a sample from the distribution of state at a certain time point is available. Typical examples include data sets collected for DNA evolution, social gathering, density in control problems, and bird migration, during the evolution of which it is impossible to follow an individual inter-temporally. In those applications, some observed individuals at one time point may be un-observable at the next time spot, or when the individual identities are blocked or unavailable due to various technical and ethical reasons. Rather than inferring the exact information for each individual, the main objective of learning dynamics in aggregate data is to recover and predict the evolution of distribution of all individuals together. **Trajectory data**, in contrast, is a kind of data that we are able to acquire the information of each individual all the time. Although some studies also considered the case that partial trajectories are missing, the identities of those individuals, whenever they are observable, are always assumed available. For example, stock price, weather, customer behaviors and most training data sets for computer vision and natural language processing are considered as trajectory data. There are many existing models to learn dynamics of full-trajectory data. Typical ones include Hamiltonian neural networks [102], Hidden Markov Model (HMM) [103, 104], Kalman Filter (KF) [105, 106, 107] and Particle Filter (PF) [108, 109], as well as the models built upon HMM, KF and PF [110, 111, 112, 113]. They require full trajectories of each individual, which may not be applicable in the aggregate data situations. On the other side, only a few methods are proposed on aggregated data in the recent learn-

ing literature. In the work of [114], authors assumed that the hidden dynamic of particles follows a stochastic differential equation (SDE), in particular, they used a recurrent neural network to parameterize the drift term. Furthermore, [115] improved traditional HMM model by using an SDE to describe the evolving process of hidden states and [116] updated HMM parameters through aggregate observations.

We propose to learn the dynamics of density through the weak form of Fokker Planck Equation (FPE), which is a parabolic partial differential equation (PDE) governing many dynamical systems subject to random noise perturbations, including the typical SDE models in existing studies. Our learning is accomplished by minimizing the Wasserstein distance between predicted distribution given by FPE and the empirical distribution from data samples. Meanwhile we utilize neural networks to handle higher dimensional cases. More importantly, by leveraging the framework of Wasserstein Generative Adversarial Network (WGAN) [7], our model is capable of approximating the distribution of samples at different time points without solving the SDE or FPE. More specifically, we treat the drift coefficient, the goal of learning, in the FPE as a generator, and the test function in the weak form of FPE as a discriminator. In other words, our method can also be regarded as a data-driven method to estimate transport coefficient in FPE, which corresponds to the drift terms in SDEs. Additionally, though we treat diffusion term as a constant in our model, it is straightforward to generalize it to be a neural network as well, which can be an extension of this work. We would like to mention that several methods of solving SDE and FPE [117, 118, 119] adopt opposite ways to our method, they utilize neural networks to estimate the distribution $P(x, t)$ with given drift and diffusion terms.

In conclusion, our contributions are: 1) We develop an algorithm that learns the drift term of a SDE via minimizing the Wasserstein discrepancy between the observed aggregate data and our generated data. 2) By leveraging a *weak* form of FPE, we are able to compute the Wasserstein distance directly without solving the FPE. 3) Finally, we demonstrate the accuracy and the effectiveness of our algorithm via several synthetic and real-world

examples.

6.2 Proposed Method

6.2.1 Fokker Planck Equation for the density evolution

We assume the individuals evolve in a pattern in the space $\mathbb{R}^D$ as shown in Figure 6.1. One example satisfying such process is the stochastic differential equation(SDE), which is also known as the Itô process [120]: $d\boldsymbol{X}_t = g(\boldsymbol{X}_t, t)dt + \sigma d\boldsymbol{W}_t$. Here $d\boldsymbol{X}_t$ represents an infinitesimal change of $\{\boldsymbol{X}_t\}$ along with time increment dt, $g(\cdot, t) = (g^1(\cdot, t), ..., g^D(\cdot, t))^T$ is the drift term (drifting vector field) that drives the dynamics of SDE, σ is the diffusion constant, $\{\boldsymbol{W}_t\}$ is the standard Brownian Motion.

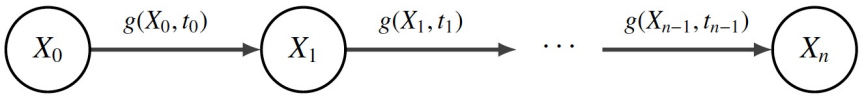


Figure 6.1: State model of the stochastic process X_t

The probability density of $\{\boldsymbol{X}_t\}$ is governed by the Fokker Planck Equation(FPE) [121]:

Lemma 6.2.1. *Suppose $\{\boldsymbol{X}_t\}$ solves the SDE $d\boldsymbol{X}_t = g(\boldsymbol{X}_t, t)dt + \sigma d\boldsymbol{W}_t$, denote $p(\cdot, t)$ as the probability density of the random variable $\boldsymbol{X}_t$. Then $p(x, t)$ solves the following equation:*

$$\frac{\partial p(\boldsymbol{x}, t)}{\partial t} = \sum_{i=1}^{D} -\frac{\partial}{\partial x_i}\left[g^i(\boldsymbol{x}, t)p(\boldsymbol{x}, t)\right] + \frac{1}{2}\sigma^2 \sum_{i=1}^{D} \frac{\partial^2}{\partial {x_i}^2} p(\boldsymbol{x}, t). \tag{6.1}$$

As a linear evolution PDE, FPE describes the evolution of density function of the stochastic process driven by a SDE. Due to this reason, FPE plays a crucial role in stochastic calculus, statistical physics and modeling [122, 123, 124]. Its importance is also drawing

more attention among statistic and machine learning communities [125, 126, 127]. In this chapter, we utilize the weak form of FPE as a basis to study hidden dynamics of the time evolving aggregated data without solving FPE.

Our task can be described as: assume that the individuals evolve with the process indicated by Figure 6.1, which can be simulated by Itô process. Then given observations $\boldsymbol{x}_t$ along time axis, we aim to recover the drift coefficient $g(\boldsymbol{x}, t)$ in FPE, and thus we are able to recover and predict the density evolution of such dynamic. For simplicity we treat $g(\boldsymbol{x}, t)$ as a function uncorrelated to time t, namely, $g(\boldsymbol{x}, t) = g(\boldsymbol{x})$. Notice that though evolving process of individuals can be simulated by Itô process, in reality since we lose identity information of individuals, the observed data become aggregate data, thus we need a new way other than traditional methods to study the swarm's distribution.

We also remark that in the work of [114], based on JKO theorem, they utilized RNN to approximate potential function and measure Sinkhorn distance (an approximation to Wasserstein distance). In our work, we assume that the density follows Fokker Planck equation, but we don't solve it directly. Instead, we take the weak form of Fokker Planck equation and compute everything in sample form, which coincides with a similar form of WGAN at the observations. Particularly, we treat FPE as the dynamic regularizer for the marginal fitting problem, therefore is fundamentally different from previous methods. As a byproduct, our numerical scheme allows to freely choose the time step Δt, which is not restricted to the given time stamp of observations. Δt is used to control the error bound.

6.2.2 Weak Form of Fokker Planck Equation

Given FPE stated in Lemma 6.2.1, if we multiply a test function $f \in H_0^1(\mathbb{R}^D)$ on both sides of the FPE, where $H_0^1(\mathbb{R}^D)$ denote the Sobolev space. Integration on both sides:

$$\int \frac{\partial p}{\partial t} f(\boldsymbol{x}) d\boldsymbol{x} = \int \sum_{i=1}^{D} -\frac{\partial}{\partial x_i} \left[g^i(\boldsymbol{x}) p(\boldsymbol{x}, t) \right] f(\boldsymbol{x}) d\boldsymbol{x} + \frac{1}{2}\sigma^2 \int \sum_{i=1}^{D} \frac{\partial^2}{\partial {x_i}^2} p(\boldsymbol{x}, t) f(\boldsymbol{x}) d\boldsymbol{x},$$

$$\int \frac{\partial p}{\partial t} f(\boldsymbol{x}) d\boldsymbol{x} = \int \sum_{i=1}^{D} g^i(\boldsymbol{x}) \frac{\partial}{\partial x_i} f(\boldsymbol{x}) p(\boldsymbol{x}, t) d\boldsymbol{x} + \frac{1}{2}\sigma^2 \int \sum_{i=1}^{D} \frac{\partial^2}{\partial {x_i}^2} f(\boldsymbol{x}) p(\boldsymbol{x}, t) d\boldsymbol{x}.$$

The first advantage of weak solution is that the solution of a PDE usually requires strong regularity and thus may not exist in the classical sense for a certain group of equations, however, the weak solution has fewer regularity requirements and thus their existence are guaranteed for a much larger classes of equations. The second advantage is that the weak formulation may provide new perspectives for numerically solving PDEs [128, 129, 130].

Suppose the observed samples at time points t_{m-1} and t_m follow the true densities $\hat{p}(\cdot, t_{m-1})$ and $\hat{p}(\cdot, t_m)$ respectively. Let's consider the following SDE:

$$\begin{aligned} & d\tilde{\boldsymbol{X}}_t = g_\omega(\tilde{\boldsymbol{X}}_t)dt + \sigma d\boldsymbol{W}_t, \\ & \text{where} \quad t_{m-1} \le t \le t_m, \quad \tilde{\boldsymbol{X}}_{t_{m-1}} \sim \hat{p}(\cdot, t_{m-1}). \end{aligned} \tag{6.2}$$

Here g_ω is an approximation to the real drift term g. In our research, we treat g_ω as a neural network with parameters ω. Stochastic process $\tilde{\boldsymbol{X}}_t$ has a density function, denoted by $\tilde{p}(\cdot, t)$, which is different from the observed density. Hence, it is natural to compute and minimize the discrepancy between the approximated density $\tilde{p}(\cdot, t_m)$ and true density $\hat{p}(\cdot, t_m)$, within which we optimize g_ω and thus recover the true drift term g.

In our research, we choose the Wasserstein-1 distance as our discrepancy function [6] [7]. Applying Kantorovich-Rubinstein duality [6] leads to $W_1(\hat{p}(\cdot, t_m), \tilde{p}(\cdot, t_m)) =$

$$\sup_{\|\nabla f\| \le 1} \left\{ \mathbb{E}_{\boldsymbol{x}_r \sim \hat{p}(\boldsymbol{x}, t_m)}[f(\boldsymbol{x}_r)] - \mathbb{E}_{\boldsymbol{x}_g \sim \tilde{p}(\boldsymbol{x}, t_m)}[f(\boldsymbol{x}_g)] \right\}.$$

The first term $\mathbb{E}_{\boldsymbol{x}_r \sim \hat{p}(\boldsymbol{x}, t_m)}[f(\boldsymbol{x}_r)]$ can be conveniently computed by Monte-Carlo method since we are already provided with the real data points $\boldsymbol{x}_r \sim \hat{p}(\cdot, t_m)$. To evaluate the sec-

ond term, we first approximate $\tilde{p}(\cdot, t_m)$ by trapezoidal rule [131]: $\tilde{p}(\boldsymbol{x}, t_m) \approx$

$$\hat{p}(\boldsymbol{x}, t_{m-1}) + \frac{\Delta t}{2}\left(\frac{\partial \hat{p}(\boldsymbol{x}, t_{m-1})}{\partial t} + \frac{\partial \tilde{p}(\boldsymbol{x}, t_m)}{\partial t}\right), \tag{6.3}$$

where $\Delta t = t_m - t_{m-1}$. Then we compute

$$\begin{aligned}&\mathbb{E}_{\boldsymbol{x}_g \sim \tilde{p}(\cdot, t_m)}[f(\boldsymbol{x}_g)] \approx \int f(\boldsymbol{x})\hat{p}(\boldsymbol{x}, t_{m-1})d\boldsymbol{x} + \\ &\frac{\Delta t}{2}\left(\int \frac{\partial \hat{p}(\boldsymbol{x}, t_{m-1})}{\partial t} f(\boldsymbol{x})d\boldsymbol{x} + \int \frac{\partial \tilde{p}(\boldsymbol{x}, t_m)}{\partial t} f(\boldsymbol{x})d\boldsymbol{x}\right).\end{aligned} \tag{6.4}$$

In the above Equation (6.4), the second and the third term on the right-hand side can be reformulated via the weak form of FPE. This gives us a new formulation for $W_1(\hat{p}(\cdot, t_m), \tilde{p}(\cdot, t_m))$, which can by computed by using Monte-Carlo method. In fact, the first and the second terms in (6.4) can be directly computed via data points from $\hat{p}(\cdot, t_{m-1})$. For the third term, we need to generate samples from $\tilde{p}(\cdot, t_m)$. To achieve this, we apply Euler-Maruyama scheme [132] to SDE (6.2) in order to acquire our desired samples $\tilde{\boldsymbol{x}}_{t_m}$:

$$\begin{aligned}&\tilde{\boldsymbol{x}}_{t_m} = \hat{\boldsymbol{x}}_{t_{m-1}} + g_\omega(\hat{\boldsymbol{x}}_{t_{m-1}})\Delta t + \sigma\sqrt{\Delta t}\boldsymbol{z}, \\ &\text{where} \quad \boldsymbol{z} \sim \mathcal{N}(0, I), \quad \hat{\boldsymbol{x}}_{t_{m-1}} \sim \hat{p}(\cdot, t_{m-1}).\end{aligned} \tag{6.5}$$

Here $\mathcal{N}(0, I)$ is the standard Gaussian distribution on $\mathbb{R}^D$. Now we summarize these results in Proposition 1:

Proposition 6.2.1. *For a set of points $X = \{\boldsymbol{x}^{(1)}, ..., \boldsymbol{x}^{(N)}\}$ in $\mathbb{R}^D$. We denote $\mathcal{F}_f(X)$ as*

$$\frac{1}{N}\sum_{k=1}^{N}\left(\sum_{i=1}^{D} g_\omega^i(\boldsymbol{x}^{(k)})\frac{\partial}{\partial x_i} f(\boldsymbol{x}^{(k)}) + \frac{1}{2}\sigma^2 \sum_{i=1}^{D}\frac{\partial^2}{\partial x_i^2} f(\boldsymbol{x}^{(k)})\right),$$

then at time point t_m, the Wasserstein distance between $\hat{p}(\cdot, t_m)$ and $\tilde{p}(\cdot, t_m)$ can be approx-

imated by

$$W_1(\hat{p}(\cdot,t_m),\tilde{p}(\cdot,t_m)) \approx \sup_{\|\nabla f\|\le 1}\Bigg\{\frac{1}{N}\sum_{k=1}^{N} f(\hat{\boldsymbol{x}}_{t_m}^{(k)}) - \frac{1}{N}\sum_{k=1}^{N} f(\hat{\boldsymbol{x}}_{t_{m-1}}^{(k)}) - \frac{\Delta t}{2}\Big(\mathcal{F}_f(\hat{X}_{m-1}) + \mathcal{F}_f(\tilde{X}_m)\Big)\Bigg\}.$$

Here $\{\hat{\boldsymbol{x}}_{t_{m-1}}^{(k)}\} \sim \hat{p}(\cdot,t_{m-1})$, $\{\hat{\boldsymbol{x}}_{t_m}^{(k)}\} \sim \hat{p}(\cdot,t_m)$. *We denote* $\hat{X}_{m-1} = \{\hat{\boldsymbol{x}}_{t_{m-1}}^{(1)},...,\hat{\boldsymbol{x}}_{t_{m-1}}^{(N)}\}$, $\tilde{X}_m = \{\tilde{\boldsymbol{x}}_{t_m}^{(1)},...,\tilde{\boldsymbol{x}}_{t_m}^{(N)}\}$, *where each* $\tilde{\boldsymbol{x}}_{t_m}^{(k)}$ *is computed by Euler-Maruyama scheme.*

Proof. Suppose $\hat{x}_{t_m}^{(k)}$ and $\hat{x}_{t_{m-1}}^{(k)}$ are our observed samples at t_m and t_{m-1} respectively, then expectations could be approximated by

$$\mathbb{E}_{\boldsymbol{x}\sim\hat{p}(\boldsymbol{x},t_m)}[f(\boldsymbol{x})] = \int f(\boldsymbol{x})\hat{p}(\boldsymbol{x},t_m)d\boldsymbol{x} \approx \frac{1}{N}\sum_{k=1}^{N} f(\hat{x}_{t_m}^{(k)}), \tag{6.6}$$

$$\begin{aligned}\mathbb{E}_{\boldsymbol{x}\sim\tilde{p}(\boldsymbol{x},t_m)}[f(\boldsymbol{x})] &= \int f(\boldsymbol{x})\tilde{p}(\boldsymbol{x},t_m)d\boldsymbol{x} = \int f(\boldsymbol{x})\left[\hat{p}(\boldsymbol{x},t_{m-1}) + \int_{t_{m-1}}^{t_m}\frac{\partial p(\boldsymbol{x},\tau)}{\partial t}d\tau\right]d\boldsymbol{x}\\ &= \int f(\boldsymbol{x})\hat{p}(\boldsymbol{x},t_{m-1})d\boldsymbol{x} + \int f(\boldsymbol{x})\int_{t_{m-1}}^{t_m}\frac{\partial p(\boldsymbol{x},\tau)}{\partial t}d\tau d\boldsymbol{x}\\ &\approx \frac{1}{N}\sum_{k=1}^{N} f(\hat{\boldsymbol{x}}_{t_{m-1}}^{(k)}) + I,\end{aligned} \tag{6.7}$$

$$I = \int f(\boldsymbol{x})\int_{t_{m-1}}^{t_m}\left\{-\sum_{i=1}^{D}\frac{\partial}{\partial x_i}\left[g_\omega^i(\boldsymbol{x})p(\boldsymbol{x},\tau)\right] + \frac{1}{2}\sigma^2\sum_{i=1}^{D}\frac{\partial^2}{\partial x_i^2}p(\boldsymbol{x},\tau)\right\}d\tau d\boldsymbol{x}. \tag{6.8}$$

Then for the second term I above, it is difficult to calculate directly, but we can use

integration by parts to rewrite I as

$$
\begin{aligned}
I &= \int_{t_{m-1}}^{t_m} \int \left[\sum_{i=1}^{D} -f(\boldsymbol{x}) \frac{\partial}{\partial x_i} g_\omega^i(\boldsymbol{x}) p(\boldsymbol{x}, \tau) + \frac{1}{2}\sigma^2 \sum_{i=1}^{D} f(\boldsymbol{x}) \frac{\partial^2}{\partial x_i^2} p(\boldsymbol{x}, \tau) \right] d\boldsymbol{x} d\tau \\
&= \int_{t_{m-1}}^{t_m} \int \left[\sum_{i=1}^{D} g_\omega^i(\boldsymbol{x}) p(\boldsymbol{x}, \tau) \frac{\partial}{\partial x_i} f(\boldsymbol{x}) + \frac{1}{2}\sigma^2 \sum_{i=1}^{D} p(\boldsymbol{x}, \tau) \frac{\partial^2}{\partial x_i^2} f(\boldsymbol{x}) \right] d\boldsymbol{x} d\tau \\
&= \int_{t_{m-1}}^{t_m} \left(\mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x},\tau)} \left[\sum_{i=1}^{D} g_\omega^i(\boldsymbol{x}) \frac{\partial}{\partial x_i} f(\boldsymbol{x}) \right] + \mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x},\tau)} \left[\frac{1}{2}\sigma^2 \sum_{i=1}^{D} \frac{\partial^2}{\partial x_i^2} f(\boldsymbol{x}) \right] \right) d\tau \\
&\approx \int_{t_{m-1}}^{t_m} \frac{1}{N} \sum_{k=1}^{N} \left(\sum_{i=1}^{D} g_\omega^i(x^{(k)}) \frac{\partial}{\partial x_i} f(x^{(k)}) + \frac{1}{2}\sigma^2 \sum_{i=1}^{D} \frac{\partial^2}{\partial x_i^2} f(x^{(k)}) \right) d\tau. \qquad (6.9)
\end{aligned}
$$

To approximate the integral from t_{m-1} to t_m, we adopt trapezoid rule, then we could rewrite the expectation in Equation (6.7) as

$$
\begin{aligned}
\mathbb{E}_{\boldsymbol{x}\sim\tilde{p}(\boldsymbol{x},t_m)}[f(\boldsymbol{x})] \approx & \frac{1}{N} \sum_{k=1}^{N} f(\hat{x}_{t_{m-1}}^{(k)}) \\
& + \frac{\Delta t}{2} \left[\frac{1}{N} \sum_{k=1}^{N} \left(\sum_{i=1}^{D} g_\omega^i(\hat{x}_{t_{m-1}}^{(k)}) \frac{\partial}{\partial x_i} f(\hat{x}_{t_{m-1}}^{(k)}) + \frac{1}{2}\sigma^2 \sum_{i=1}^{D} \frac{\partial^2}{\partial x_i^2} f(\hat{x}_{t_{m-1}}^{(k)}) \right) \right. \\
& \left. + \frac{1}{N} \sum_{k=1}^{N} \left(\sum_{i=1}^{D} g_\omega^i(\tilde{x}_{t_m}^{(k)}) \frac{\partial}{\partial x} f(\tilde{x}_{t_m}^{(k)}) + \frac{1}{2}\sigma^2 \sum_{i=1}^{D} \frac{\partial^2}{\partial x_i^2} f(\tilde{x}_{t_m}^{(k)}) \right) \right] \\
= & \frac{1}{N} \sum_{k=1}^{N} f(\hat{x}_{t_{m-1}}^{(k)}) + \frac{\Delta t}{2} \left[\mathcal{F}_f(\hat{X}_{m-1}) + \mathcal{F}_f(\tilde{X}_m) \right]. \qquad (6.10)
\end{aligned}
$$

We subtract (6.6) by (6.10) to finish the proof. □

6.2.3 Wasserstein Distance on Time Series

In real cases, it is not realistic to observe the data at arbitrary two consecutive time nodes, especially when Δt is small. To make our model more flexible, we extend our formulation so that we are able to plug in observed data at arbitrary time points. To be more precise, suppose we observe data set $\hat{X}_{t_n} = \{\hat{\boldsymbol{x}}_{t_n}^{(1)}, ..., \hat{\boldsymbol{x}}_{t_n}^{(N)}\}$ at $J+1$ different time points $t_0, t_1, ..., t_J$. And we denote the generated data set as $\tilde{X}_{t_n} = \{\tilde{\boldsymbol{x}}_{t_n}^{(1)}, ..., \tilde{\boldsymbol{x}}_{t_n}^{(N)}\}$, here each $\tilde{\boldsymbol{x}}_{t_n}^{(\cdot)}$

is derived from the n-step Euler-Maruyama scheme:

$$\begin{aligned}&\tilde{\boldsymbol{x}}_{t_j} = \tilde{\boldsymbol{x}}_{t_{j-1}} + g_\omega(\tilde{\boldsymbol{x}}_{t_{j-1}})\Delta t + \sigma\sqrt{\Delta t}\boldsymbol{z},\\ &\text{where} \quad \boldsymbol{z} \sim \mathcal{N}(0, I), \quad 0 \le j \le n, \quad \tilde{\boldsymbol{x}}_{t_0} \sim \hat{p}(\cdot, t_0).\end{aligned} \tag{6.11}$$

Let us denote $\tilde{p}(\cdot, t)$ as the solution to FPE (6.1) with g replaced by g_ω and with initial condition $\tilde{p}(\cdot, t_0) = \hat{p}(\cdot, t_0)$, then the approximation formula for evaluating the Wasserstein distance $W_1(\hat{p}(\cdot, t_n), \tilde{p}(\cdot, t_n))$ is provided in the following proposition:

Proposition 6.2.2. *Suppose we keep all the notations defined as above, then we have the approximation:*

$$\begin{aligned}W_1(\hat{p}(\cdot, t_n), \tilde{p}(\cdot, t_n)) \approx \sup_{\|\nabla f\| \le 1} \Bigg\{ &\frac{1}{N}\sum_{k=1}^{N} f(\hat{\boldsymbol{x}}_{t_n}^{(k)}) - \frac{1}{N}\sum_{k=1}^{N} f(\hat{\boldsymbol{x}}_{t_0}^{(k)}) - \frac{\Delta t}{2}\Bigg(\mathcal{F}_f(\hat{X}_0) + \mathcal{F}_f(\tilde{X}_n)\\ &+ 2\sum_{s=1}^{n-1} \mathcal{F}_f(\tilde{X}_s)\Bigg)\Bigg\}.\end{aligned}$$

Proof. Given initial $\hat{\boldsymbol{x}}_{t_0}$, we generate $\tilde{\boldsymbol{x}}_{t_1}$, $\tilde{\boldsymbol{x}}_{t_2}$, $\tilde{\boldsymbol{x}}_{t_3}$ ... $\tilde{\boldsymbol{x}}_{t_n}$ sequentially by Euler-Maruyama scheme. Then the expectations can be rewritten as:

$$\mathbb{E}_{\boldsymbol{x}\sim\hat{p}(\boldsymbol{x}, t_n)}[f(\boldsymbol{x})] = \int f(\boldsymbol{x})\hat{p}(\boldsymbol{x}, t_n)d\boldsymbol{x} \approx \frac{1}{N}\sum_{k=1}^{N} f(\hat{x}_{t_n}^{(k)}) \tag{6.12}$$

$$\begin{aligned}\mathbb{E}_{\boldsymbol{x}\sim\tilde{p}(\boldsymbol{x},t_n)}[f(\boldsymbol{x})] \approx & \frac{1}{N}\sum_{k=1}^{N} f(\hat{x}_{t_0}^{(k)}) \\ & + \int_{t_0}^{t_1} \frac{1}{N}\sum_{k=1}^{N}\left[\sum_{i=1}^{D} g_\omega^i(x^{(k)})\frac{\partial}{\partial x_i} f(x^{(k)}) + \frac{1}{2}\sigma^2 \sum_{i=1}^{D}\frac{\partial^2}{\partial x_i \partial x_j} f(x^{(k)})\right] d\tau \\ & + \int_{t_1}^{t_2} \frac{1}{N}\sum_{k=1}^{N}\left[\sum_{i=1}^{D} g_\omega^i(x^{(k)})\frac{\partial}{\partial x_i} f(x^{(k)}) + \frac{1}{2}\sigma^2 \sum_{i=1}^{D}\frac{\partial^2}{\partial x_i^2} f(x^{(k)})\right] d\tau + ... \\ & + \int_{t_{n-1}}^{t_n} \frac{1}{N}\sum_{k=1}^{N}\left[\sum_{i=1}^{n} g_\omega^i(x^{(k)})\frac{\partial}{\partial x_i} f(x^{(k)}) + \frac{1}{2}\sigma^2 \sum_{i=1}^{n}\frac{\partial^2}{\partial x_i^2} f(x^{(k)})\right] d\tau\end{aligned} \tag{6.13}$$

which is

$$\begin{aligned}\mathbb{E}_{\boldsymbol{x}\sim\tilde{p}(\boldsymbol{x},t_n)}[f(\boldsymbol{x})] \approx & \frac{1}{N}\sum_{k=1}^{N} f(\hat{x}_{t_0}^{(k)}) + \frac{\Delta t}{2}\left[\mathcal{F}_f(\hat{X}_0) + \mathcal{F}_f(\tilde{X}_1)\right] + \frac{\Delta t}{2}\left[\mathcal{F}_f(\tilde{X}_1) + \mathcal{F}_f(\tilde{X}_2)\right] \\ & + ... + \frac{\Delta t}{2}\left[\mathcal{F}_f(\tilde{X}_{n-1}) + \mathcal{F}_f(\tilde{X}_n)\right]\end{aligned} \tag{6.14}$$

Finally it comes to

$$\mathbb{E}_{\boldsymbol{x}\sim\tilde{p}(\boldsymbol{x},t_n)}[f(\boldsymbol{x})] \approx \frac{1}{N}\sum_{k=1}^{N} f(\hat{x}_{t_0}^{(k)}) + \frac{\Delta t}{2}\left(\mathcal{F}_f(\hat{X}_0) + \mathcal{F}_f(\tilde{X}_n) + 2\sum_{s=1}^{n-1}\mathcal{F}_f(\tilde{X}_s)\right) \tag{6.15}$$

We subtract (6.12) by (6.15) to finish the proof. □

Minimizing the Objective Function: Base on Proposition 6.2.2, we obtain objective function by summing up the accumulated Wasserstein distances among J observations along the time axis. Thus, our goal is to minimize the following objection function:

$$\begin{aligned}\min_{g_\omega}\Bigg\{\sum_{n=1}^{J}\sup_{\|\nabla f_n\|\le 1}\Bigg\{ & \frac{1}{N}\sum_{k=1}^{N} f_n(\hat{\boldsymbol{x}}_{t_n}^{(k)}) - \frac{1}{N}\sum_{k=1}^{N} f_n(\hat{\boldsymbol{x}}_{t_0}^{(k)}) \\ & - \frac{\Delta t}{2}\left(\mathcal{F}_{f_n}(\hat{X}_0) + \mathcal{F}_{f_n}(\tilde{X}_n) + 2\sum_{s=1}^{n-1}\mathcal{F}_{f_n}(\tilde{X}_s)\right)\Bigg\}\Bigg\}.\end{aligned}$$

Notice that since we have observations on J distinct time points, for each time point we

Algorithm 6 Fokker Planck Process Algorithm

Require: Initialize f_{θ_n} $(1 \leq n \leq J)$, g_ω

Require: Set ϵ_{f_n} as the inner loop learning rate for f_{θ_n} and ϵ_g as the outer loop learning rate for g_ω

for # training iterations **do**
  for k steps **do**
    for observed time t_s in $\{t_1, ..., t_J\}$ **do**
      Compute the generated data set $\tilde{X}_{t_s}$ from Euler-Maruyama scheme (6.11) for $1 \leq s \leq J$
      Acquire data sets $\hat{X}_{t_s} = \{\hat{\boldsymbol{x}}_{t_s}^{(1)}, ..., \hat{\boldsymbol{x}}_{t_s}^{(N)}\}$ from real distribution $\hat{p}(\cdot, t_s)$ for $1 \leq s \leq J$
    end for
    For each dual function f_{θ_n}, compute: $\mathcal{F}_n = \mathcal{F}_{f_{\theta_n}}(\hat{X}_{t_0}) + \mathcal{F}_{f_{\theta_n}}(\tilde{X}_{t_n}) + 2\sum_{s=1}^{n-1} \mathcal{F}_{f_{\theta_n}}(\tilde{X}_{t_s})$
    Update each f_{θ_n} by:
    $$\theta_n \leftarrow \theta_n + \epsilon_{f_n} \nabla_\theta \left(\frac{1}{N} \sum_{k=1}^{N} f_{\theta_n}(\hat{\boldsymbol{x}}_{t_n}^{(k)}) - \frac{1}{N} \sum_{k=1}^{N} f_{\theta_n}(\hat{\boldsymbol{x}}_{t_0}^{(k)}) - \frac{\Delta t}{2} \mathcal{F}_n \right)$$
  end for
  Update g_ω by:
  $$\omega \leftarrow \omega - \epsilon_g \nabla_\omega \left(\sum_{n=1}^{J} \left(\frac{1}{N} f_{\theta_n}(\hat{\boldsymbol{x}}_{t_n}^{(k)}) - \frac{1}{N} f_{\theta_n}(\hat{\boldsymbol{x}}_{t_0}^{(k)}) - \frac{\Delta t}{2} \mathcal{F}_n \right) \right)$$
end for

compute Wasserstein distance with the help of the dual function f_n, thus we involve J test functions in total. In our actual implementation, we will choose these dual functions as neural networks. We call our algorithm Fokker Planck Process(FPP), the entire procedure is shown in Algorithm 6. We also provide an error analysis in Appendix.

Remark 6. *When the time interval $\Delta t = t_j - t_i$ between two observations at X_i and $X_j (i < j)$ is large. In order to guarantee the accuracy of $\tilde{X}_s$, we can separate Δt into multiple smaller intervals, namely, $\Delta t = Kh$, where K the number of intervals and h is the interval length. Then we evaluate* (6.11) *on the finer meshes to obtain more accurate samples $\{\tilde{x}_s^{(1)}, ..., \tilde{x}_s^{(N)}\}$ at specific time s.*

Remark 7. *The drift function recovered by our framework may not be unique, see Section 4 for more details.*

6.3 Experiments

In this section, we evaluate our model on various synthetic and realistic data sets by employing Algorithm 6. We generate samples $\tilde{\boldsymbol{x}}_t$ and make all predictions base on Equation (6.5) starting with $\hat{\boldsymbol{x}}_0$.

Baselines: We compare our model with two recently proposed methods. One model (NN) adopts recurrent neural network(RNN) to learn dynamics directly from observations of aggregate data [114]. The other one model (LEGEND) learns dynamics in a HMM framework [115]. The baselines in our experiments are two typical representatives that have state-of-the-art performance on learning aggregate data. Furthermore, though we simulate the evolving process of the data as a SDE, which is on the same track with NN, as mentioned before, NN trains its RNN via optimizing Sinkhorn distance [133], our model starts with a view of weak form of PDE, focuses more on WGAN framework and easier computation.

6.3.1 Synthetic Data

We first evaluate our model on three synthetic data sets which are generated by three artificial dynamics: **Synthetic-1**, **Synthetic-2** and **Synthetic-3**.

Experiment Setup: In all synthetic data experiments, we set the drift term g and the discriminator f as two simple fully-connected networks. The g network has one hidden layer and the f network has three hidden layers. Each layer has 32 nodes for both g and f. The only one activation function we choose is Tanh. Notice that since we need to calculate $\frac{\partial^2 f}{\partial x^2}$, the activation function of f must be twice differentiable to avoid loss of weight gradient. In terms of training process, we use the Adam optimizer [55] with learning rate 10^{-4}. Furthermore, we use spectral normalization to realize $\|\nabla f\| \leq 1$[34]. We initialize the weights with Xavier initialization[99] and train our model by Algorithm 1. We set the data size at each time point is $N = 2000$, treat 1200 data points as the training

set and the other 800 data points as the test set, Δt is set to be 0.01.

Synthetic-1:

$$\hat{\boldsymbol{x}}_0 \sim \mathcal{N}(0, \boldsymbol{\Sigma}_0),$$
$$\hat{\boldsymbol{x}}_{t+\Delta t} = \hat{\boldsymbol{x}}_t - (\boldsymbol{A}\hat{\boldsymbol{x}}_t + \boldsymbol{b})\Delta t + \sigma\sqrt{\Delta t}\mathcal{N}(0, 1).$$

In Synthetic-1, the data is following a simple linear dynamic, we set $\boldsymbol{A} = [(4, 0), (0, 1)]$, $\boldsymbol{b} = [-12,\ -12]^T$, $\sigma = 1$, $\boldsymbol{\Sigma}_0 = \boldsymbol{I_2}$. We utilize true $\boldsymbol{x}_0$, $\boldsymbol{x}_{20}$ and $\boldsymbol{x}_{200}$ in training process and predict the distributions of $\boldsymbol{x}_{10}$, $\boldsymbol{x}_{50}$ and $\boldsymbol{x}_{500}$. As visualized in Figure 6.2, from (a) to (c), the generated data(blue) covers all areas of ground truth(red), the original Gaussian distribution converges to the target Gaussian distribution as we expect.

Synthetic-2:

$$\hat{\boldsymbol{x}}_0 \sim \mathcal{N}(0, \boldsymbol{\Sigma}_0), \quad \hat{\boldsymbol{x}}_{t+\Delta t} = \hat{\boldsymbol{x}}_t - \boldsymbol{G}\Delta t + \sigma\sqrt{\Delta t}\mathcal{N}(0, 1),$$

where $\boldsymbol{G}$ is given as following.

Definition of G in Synthetic-2

$$\boldsymbol{G}_{11} = \frac{1}{\sigma_1}\frac{N_1}{N_1 + N_2}(\hat{x}_t^1 - \mu_{11}) + \frac{1}{\sigma_2}\frac{N_2}{N_1 + N_2}(\hat{x}_t^1 - \mu_{21})$$
$$\boldsymbol{G}_{22} = \frac{1}{\sigma_1}\frac{N_1}{N_1 + N_2}(\hat{x}_t^2 - \mu_{12}) + \frac{1}{\sigma_2}\frac{N_2}{N_1 + N_2}(\hat{x}_t^2 - \mu_{22})$$
$$N_1 = \frac{1}{\sqrt{2\pi}\sigma_1}\exp\left(-\frac{(\hat{x}_t^1 - \mu_{11})^2}{2\sigma_1^2} - \frac{(\hat{x}_t^1 - \mu_{12})^2}{2\sigma_1^2}\right)$$
$$N_2 = \frac{1}{\sqrt{2\pi}\sigma_2}\exp\left(-\frac{(\hat{x}_t^2 - \mu_{21})^2}{2\sigma_2^2} - \frac{(\hat{x}_t^2 - \mu_{22})^2}{2\sigma_2^2}\right)$$

In Synthetic-2, the data is following a complex nonlinear dynamic. We let $\sigma = \sigma_1 = \sigma_2 = 4$, $\mu_1 = [12, 15]^T$ and $\mu_2 = [-15, -15]^T$(defined in Appendix). We utilize true $\boldsymbol{x}_{10}$, $\boldsymbol{x}_{40}$ and $\boldsymbol{x}_{80}$ in training process and predict $\boldsymbol{x}_{30}$, $\boldsymbol{x}_{50}$ and $\boldsymbol{x}_{100}$. The results are shown in Figure 6.2, from (d) to (f), the generated data(blue) covers all areas of ground truth(red), generated

samples split and converge to a mixed Gaussian as the ground truth suggests.

Synthetic-3 (Nonlinear Van der Pol oscillator [134]:)

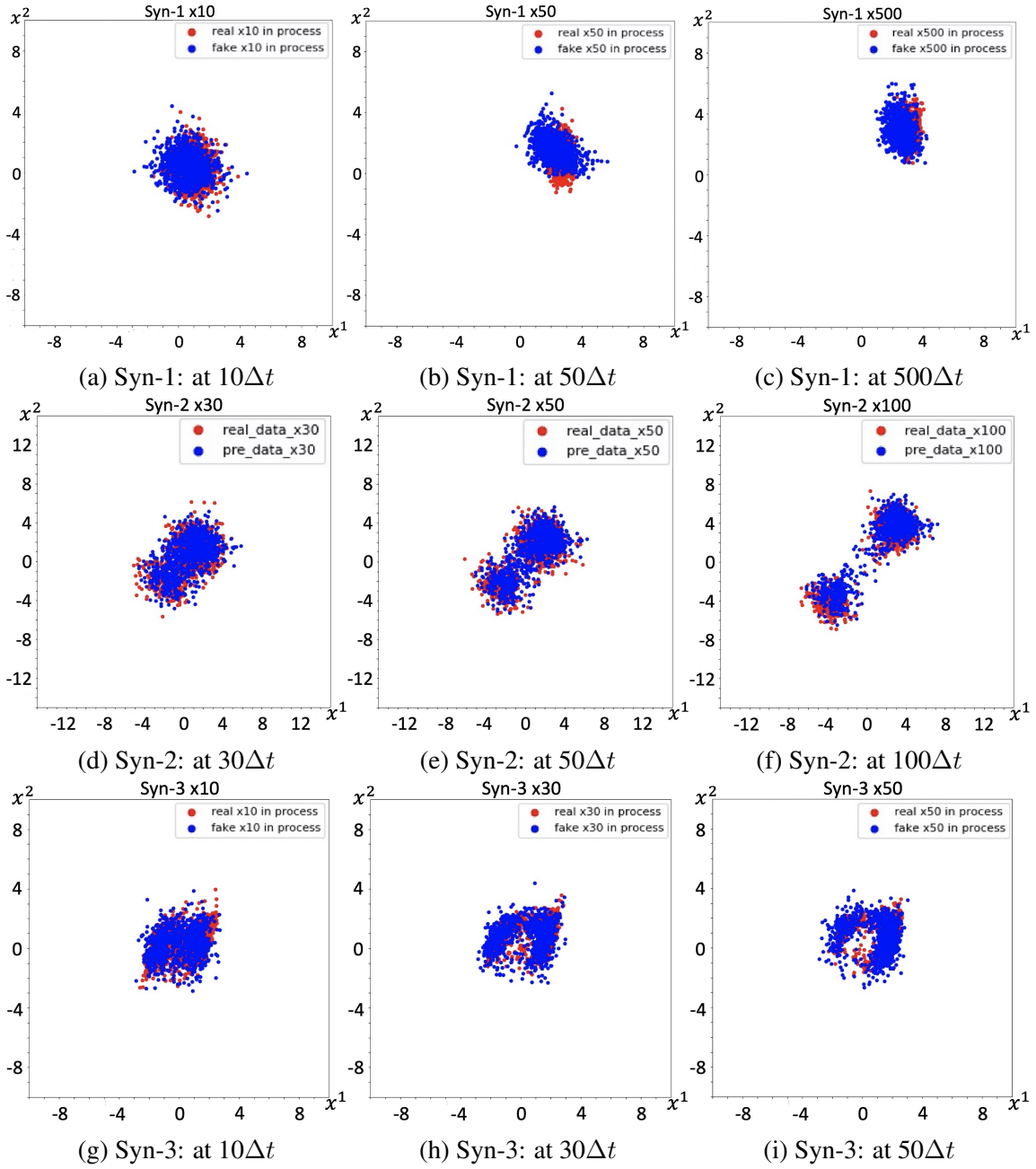
$$
\begin{aligned}
&\hat{\boldsymbol{x}}_0 \sim \mathcal{N}(0, \boldsymbol{\Sigma}_0), \\
&\hat{x}^1_{t+\Delta t} = \hat{x}^1_t + 10\left(\hat{x}^2_t - \frac{1}{3}(\hat{x}^1_t)^3 + \hat{x}^1_t\right)\Delta t + \sigma\sqrt{\Delta t}\mathcal{N}(0,1), \\
&\hat{x}^2_{t+\Delta t} = \hat{x}^2_t + 3(1 - \hat{x}^1_t)\Delta t + \sigma\sqrt{\Delta t}\mathcal{N}(0,1).
\end{aligned}
$$

In Synthetic-3, we let $\sigma = 1$ and utilize true $\boldsymbol{x}_3$, $\boldsymbol{x}_7$ and $\boldsymbol{x}_{20}$ in training process then predict the distributions of $\boldsymbol{x}_{10}$, $\boldsymbol{x}_{30}$ and $\boldsymbol{x}_{50}$. As presented in Figure 6.2, from (g) to (i), the generated data(blue) covers all areas of ground truth(red), the distributions we predict are following the true stochastic oscillator's pattern.

Remark 3: In Syn-2 and Syn-3, $\hat{x}^i_t$ represents the i-th dimension of $\hat{\boldsymbol{x}}_t$. **We further state that** in Syn-1 and Syn-3, the training data is coming from the same $\boldsymbol{x}_0$ respectively. In Syn-2 the training data is coming from different $\boldsymbol{x}_0$, namely, the training data $\boldsymbol{x}_{10}$, $\boldsymbol{x}_{40}$ and $\boldsymbol{x}_{80}$ are generated from three different sets of $\boldsymbol{x}_0$. We also consider cases in higher dimensions: D = 6 and 10. To be more precise, we couple three 2-D dynamical systems to create the 6-D dynamical system and five 2-D systems to create the 10-D example. We compare our model with the two baseline models by using Wasserstein distance as error metric for the low-dimensional (D = 2) and high-dimensional (D = 6, 10) cases. As reported in Table 6.1, our model achieves lower Wasserstein error than the two baseline models in all cases. Clearly all the drift functions in the synthetic data sets cause the change of the distributions. In Discussion Section we discuss a special case when the drift term does not change the distribution.

6.3.2 Realistic Data – RNA Sequence of Single Cell

In this section, we evaluate our model on a realistic biology data set called Single-cell RNA-seq[135], which is typically used for learning the evolvement of cell differentiation.



(a) Syn-1: at $10\Delta t$ (b) Syn-1: at $50\Delta t$ (c) Syn-1: at $500\Delta t$

(d) Syn-2: at $30\Delta t$ (e) Syn-2: at $50\Delta t$ (f) Syn-2: at $100\Delta t$

(g) Syn-3: at $10\Delta t$ (h) Syn-3: at $30\Delta t$ (i) Syn-3: at $50\Delta t$

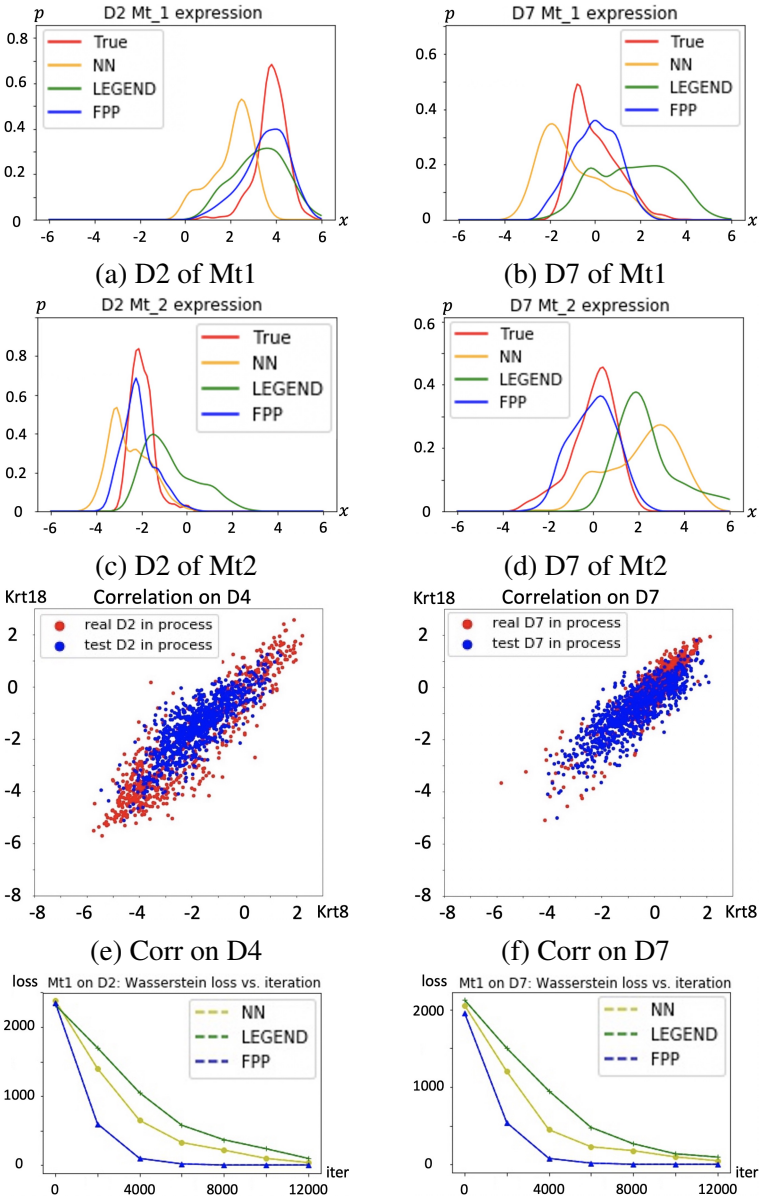
Figure 6.2: Comparison of generated data(blue) and ground truth(red) of Synthetic-1((a) to (c)), Synthetic-2((d) to (f)) and Synthetic-3((g) to (i)). In each case, it finally converges to a stationary distribution.

The cell population begins to differentiate at day 0 (D0). Single-cell RNA-seq observations are then sampled at day 0 (D0), day 2 (D2), day 4 (D4) and day 7 (D7). At each time point, the expression of 24,175 genes of several hundreds cells are measured (933, 303, 683 and 798 cells on D0, D2, D4 and D7 respectively). Notice that there is only whole group's

distribution but no trajectory information of each gene on different days. We pick 10 gene markers out of 24,175 to make a 10 dimensional data set. In the first task we treat gene expression at D0, D4 and D7 as training data to learn the hidden dynamic and predict the distribution of gene expression at D2. In the second task we train the model with gene expression at D0, D2 and D4, then predict the distribution of gene expression at D7. We plot the prediction results of two out of ten markers, i.e. Mt1 and Mt2 in Figure 6.3.

Experiment Setup: We set both f and g as fully connected three-hidden-layers neural networks, each layer has 64 nodes. The only activation function we choose is Tanh. The other setups of neural networks and training process are the same with the ones we use in Synthetic data. Notice that in realistic cases, Δt and $T/\Delta t$ become hyperparameters, here we choose $\Delta t = 0.05$, $T/\Delta t = 35$, which means the data evolves $10\Delta t$ from D0 to D2 , then $10\Delta t$ from D2 to D4 and finally $15\Delta t$ from D4 to D7. For preprocessing, we apply standard normalization procedures [136] to correct batch effects and use non-negative matrix factorization to impute missing expression levels[114, 115].

Results: As shown in Table 6.1, when compared to other baselines, our model achieves lower Wasserstein error on both Mt1 and Mt2 data, which proves that our model is capable of learning the hidden dynamics of the two studied gene expressions. In Figure 6.3 (a) to (d), we visualized the predicted distributions of the two genes. The distributions of Mt1 and Mt2 predicted by our model (curves in blue) are closer to the true distributions (curves in red) on both D2 and D7. Furthermore, our model precisely indicates the correlations between Krt8 and Krt18 on D4 and D7, as shown in Figure 6.3 (e) and (f), which also demonstrates the effectiveness of our model since closer to the true correlation represents better performance (more results in Appendix). In Figure 6.3 (g) and (h), we see the training process of our model is easier with least computation time.



(a) D2 of Mt1 (b) D7 of Mt1

(c) D2 of Mt2 (d) D7 of Mt2

(e) Corr on D4 (f) Corr on D7

(g) W-loss of Mt1, D2 (h) W-loss of Mt1, D7

Figure 6.3: (a) to (d): The performance comparisions among different models on D2 and D7 of Mt1 and Mt2. (e) and (f): True (red) and predicted (blue) correlations between Mt1(x-axis) and Mt2(y-axis) on D2 (left) and D7 (right). (g) and (h): Wasserstein loss of Mt1 on D2 and D7 vs iterations.

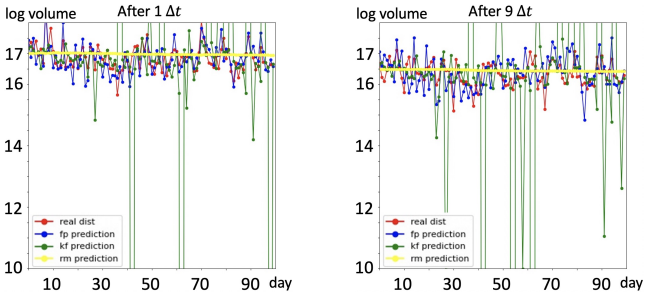
6.3.3 Realistic Data – Daily Trading Volume

In this section we would like to demonstrate the performance of our model in financial area. Trading volume is the total quantity of shares or contracts traded for specified securi-

Table 6.1: The Wasserstein error of different models on Synthetic-1/2/3 and RNA-sequence data sets.

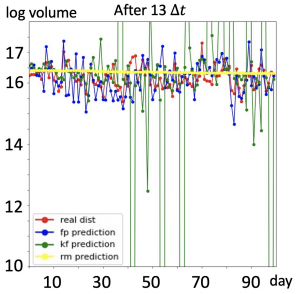
Data	Task	Dimension	NN	LEGEND	Ours
Syn–1	$\boldsymbol{x}_{50}$	2	1.37	0.44	**0.05**
		6	4.79	2.32	**0.06**
		10	9.13	2.89	**0.10**
	$\boldsymbol{x}_{500}$	2	0.84	0.18	**0.03**
		6	3.28	0.30	**0.03**
		10	8.05	1.79	**0.09**
Syn–2	$\boldsymbol{x}_{50}$	2	4.72	2.84	**0.02**
		6	6.47	5.33	**0.14**
		10	12.58	7.21	**0.22**
	$\boldsymbol{x}_{100}$	2	3.83	2.98	**0.04**
		6	8.83	3.17	**0.19**
		10	14.11	5.65	**0.32**
Syn–3	$\boldsymbol{x}_{30}$	2	4.13	1.29	**0.08**
		6	6.40	3.16	**0.17**
		10	11.76	8.53	**0.25**
	$\boldsymbol{x}_{50}$	2	3.05	0.87	**0.12**
		6	6.72	1.52	**0.16**
		10	9.81	3.55	**0.23**
RNA-Mt1	D2	10	33.86	10.28	**4.23**
	D7	10	12.69	7.21	**2.92**
RNA-Mt2	D2	10	31.45	13.32	**4.04**
	D7	10	11.58	7.89	**1.50**

ties such as stocks, bonds, options contracts, future contracts and all types of commodities. It can be measured on any type of security traded during a trading day or a specified time period. In our case, daily volume of trade is measured on stocks. Predicting traded volume is an essential component in financial research since the traded volume, as a basic component or input of other financial algorithms, tells investors the market's activity and liquidity. The data set we use is the historical traded volume of the stock "JPM". The data covers period from January 2018 to January 2020 and is obtained from Bloomberg. Each day from 14:30 to 20:55, we have 1 observation every 5 minutes, totally 78 observations everyday. Our task is described as follows: we treat historical traded volume at 14:30, 14:40, 15:05, 15:20 and 16:20, namely, $\boldsymbol{x}_0, \boldsymbol{x}_2, \boldsymbol{x}_7, \boldsymbol{x}_{10}, \boldsymbol{x}_{22}$ as training data, each time point in-

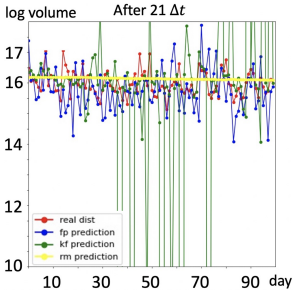


(a) 14:35

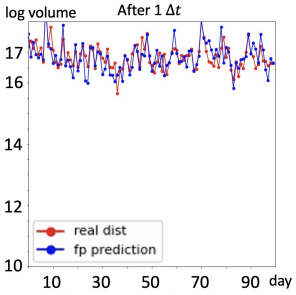
(b) 15:15



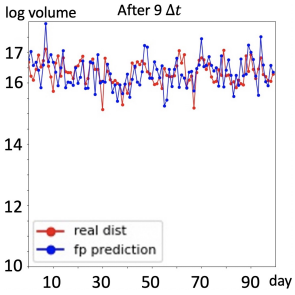
(c) 15:35



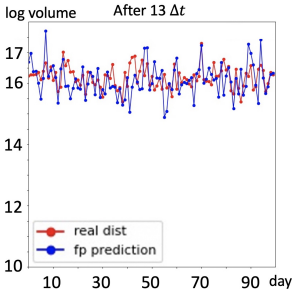
(d) 16:15



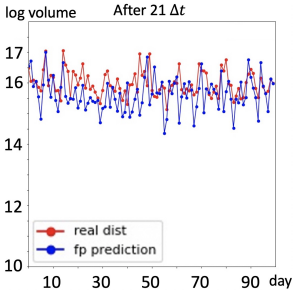
(e) 14:35



(f) 15:15



(g) 15:35



(h) 16:15

Figure 6.4: Predictions of various models, (a) to (d): Group A: predictions of our model with full trajectory, (e) to (h): Group B: predictions of our model without full trajectory.

cludes 730 samples. Then for next 100 days we predict traded volume at 14:35, 15:15, 15:35 and 16:15, namely, $\boldsymbol{x}_1, \boldsymbol{x}_9, \boldsymbol{x}_{13}, \boldsymbol{x}_{21}$. One of baselines we choose is classical rolling means(RM) method, which predicts intraday volume of a particular time interval by the

average volume traded in the same interval over the past days. The other one baseline is a kalman filter based model [137] that outperforms all available models in predicting intrady trading volume.

Experiment Setup: Following similar setup as we did for RNA data set, we utilize the same structures for neural networks here. For hyperparameters we set $\Delta t = 0.02$, $T/\Delta t = 22$, it takes one single Δt from $\boldsymbol{x}_t$ to $\boldsymbol{x}_{t+1}$. For preprocessing, we rescale data by taking natural logarithm of trading volume, which is a common way in trading volume research. We conduct experiments on two groups(A&B) to show advantages of our method, for group A we train our model on complete data set, in this case the data has full trajectory; for group B we manually delete some trajectories of the data, for instance, we randomly kick out some samples of $\boldsymbol{x}_0, \boldsymbol{x}_2, \boldsymbol{x}_7, \boldsymbol{x}_{10}, \boldsymbol{x}_{22}$ then follow the same procedures of training and prediction.

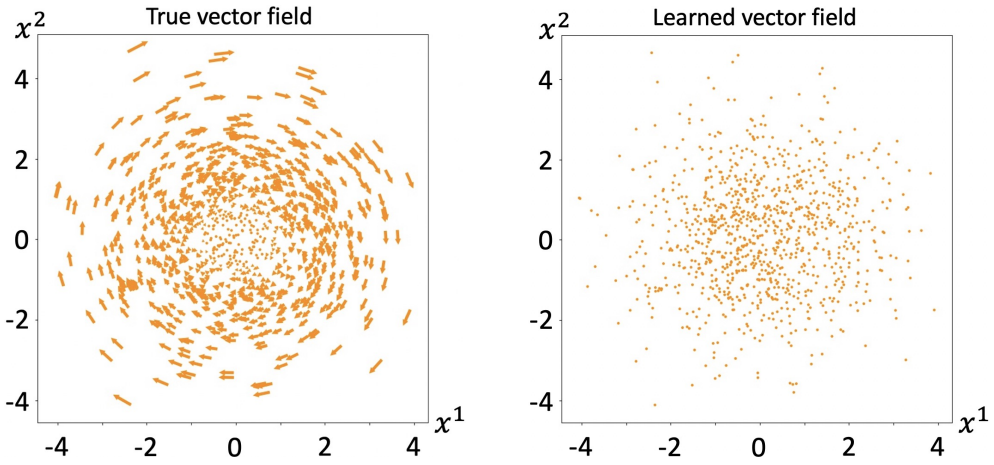
Results: We present prediction results in Figure 6.4. As shown in first four figures, RM(yellow) fails to capture the regularities of traded volume in time series, kalman filter based model(green) fails to capture noise information and make reasonable predictions, our model(blue) is able to seize the movements of traded volume and yield better predictions. With full trajectory, prediction made by RM is almost a straight line, the prediction value bouncing up and down within a very small range, thus this model cannot capture the volume movements, namely, regularities existing in the time series; prediction made by the Kalman filter based model captures the regularities better than RM model, but it fails to deal with noise component existing in the time series, thus some predictions are out of a reasonable range. Traded volume predicted by our model is closer to the real case, moreover, our model captures regularities meanwhile gives stable predictions. Furthermore, without full trajectory, Kalman filter based model fails to be applied here and RM model still fails to capture the regularities, we randomly drop half of the training samples and display predictions made by our model in last four figures of Figure 6.4, we see our model still works well.

6.4 Discussions

In this section we discuss the limitations and extension of our model.

The challenge for non-uniqueness: Mathematically it is impossible to recover the exact drift term of an SDE if we are only given the information of density evolution on certain time intervals, because there might be infinitely many drift functions to induce the same density evolution. More precisely, suppose $p(x,t)$ solves FPE (6.1), consider

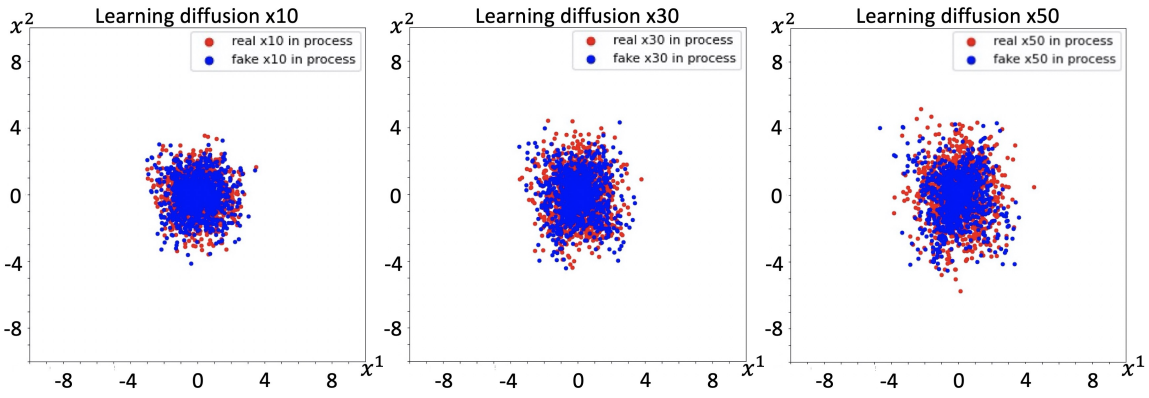
$$0 = -\sum_{i=1}^{D} \frac{\partial}{\partial x_i}(u^i(\boldsymbol{x},t)p(\boldsymbol{x},t)) + \frac{\sigma^2}{2}\sum_{i=1}^{D}\frac{\partial^2}{\partial x_i^2}p(\boldsymbol{x},t).$$



(a) True vector field (b) Learned vector field

Figure 6.5: Results of learning curl field

One can prove, under mild assumptions, that there may be infinitely many vector fields $u(\boldsymbol{x},t) = (u^1(\boldsymbol{x},t),...,u^D(\boldsymbol{x},t))$ solving above equation. Therefore the solution to FPE (6.1) with drift term $g(\boldsymbol{x},t) + u(\boldsymbol{x},t)$ is still $p(\boldsymbol{x},t)$, i.e. the vector field $u(\boldsymbol{x},t)$ never affects the density evolution of the dynamic. This illustrates that given the density evolution $p(\cdot,t)$, the solution for drift term is not necessarily unique. This clearly poses an essential difficulty of determining the exact drift term from the density. In this study, the main goal is to recover the entire density evolution (i.e. interpolate the density between observation time points) and predict how the density evolves in the future. As a result, although we



(a) Prediction at t_{10} (b) Prediction at t_{30} (c) Prediction at t_{50}

Figure 6.6: Results of learning diffusion function

cannot always acquire the exact drift term of the dynamic, we can still accurately recover and predict the density evolution. This is still meaningful and may find its application in various scientific domains.

Curl field: The drift function we showed in the synthetic experiments will apparently cause the evolution of the distribution. If the drift function is a curl, namely $g = \nabla \times \boldsymbol{F}$, then the distribution does not change, under this situation we cannot learn the density evolution since our algorithm depends on the change of the whole distribution. To demonstrate this point of view, we simulate a curl field $(y, -x)$ induced by $\boldsymbol{A} = [0, 10; -10, 0]$ on a Gaussian distribution that mean$= (0, 0)$, covariance$= [2, 0; 0, 2]$. Here we set noise part as 0. As shown in Figure 6.5, true and learned vector fields are indicated in (a) and (b) respectively. We see that the learned vectors are all "points", meaning the length of the vectors is "0", the algorithm fails to recover true vector field.

Learning diffusion function: Our framework also works for learning unknown diffusion function in the Itô process. As an extension of our work, if we approximate the diffusion function with a neural network σ_η (with parameters η), we revise the operator $\mathcal{F}$

as:

$$\mathcal{F}_f(X) = \frac{1}{N}\sum_{k=1}^{N}\Bigg(\sum_{i=1}^{D} g_\omega^i(\boldsymbol{x}^{(k)})\frac{\partial}{\partial x_i} f(\boldsymbol{x}^{(k)}) + \sum_{i=1}^{D}\Bigg(\sum_{j=1}^{D}\frac{1}{2}(\sigma_\eta^{ij}(\boldsymbol{x}^{(k)}))^2\Bigg)\frac{\partial^2}{\partial x_i^2} f(\boldsymbol{x}^{(k)})\Bigg),$$

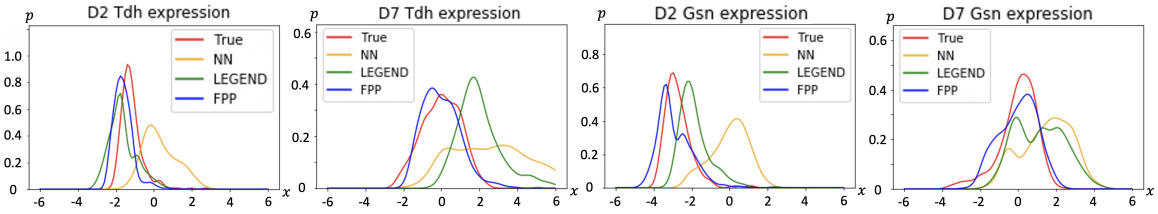
which can be derived by the same technique we used to derive Proposition 6.2.1.

We test this formulation on a synthetic data set, where we only consider diffusion influence, namely, drift term in Equation 6.2.1 is ignored. We set the ground truth of diffusion coefficient as $\boldsymbol{\sigma} = [(1,0),(0,2)]$. We design the neural network as a simple one fully connected layer with 32 nodes, then show our result in Figure 6.6, we see that the predictions(blue) follow the same patterns as the ground truth(red) does.

Future directions It worth mentioning that our proposed algorithm 6 requires the gradient with respect to the parameter ω of drift term g_ω (i.e. line 10 of Algorithm 6). Notice that each sample $\tilde{x}_{t_n}^{(k)}$ is computed from (6.11) for n steps, thus each sample $\tilde{x}_{t_n}^{(k)}$ can be treated as n compositions of drift term g_ω, which may lead to more expensive computation. However, we cann avoid direct computation of gradient ∇_ω by applying the adjoint method with Fokker-Planck equation (6.1) as the constraint [138, 139]. This is one of our future research directions. Moreover, our model can also readily handle high dimensional cases by leveraging deep neural networks. Providing more numerical analysis such as compare trapezoidal rule and Runge-Kutta method in our scheme, as well as exploring super high dimensional applications are also appealing future directions.

6.5 More experiments and proofs

6.5.1 RNA-sequence



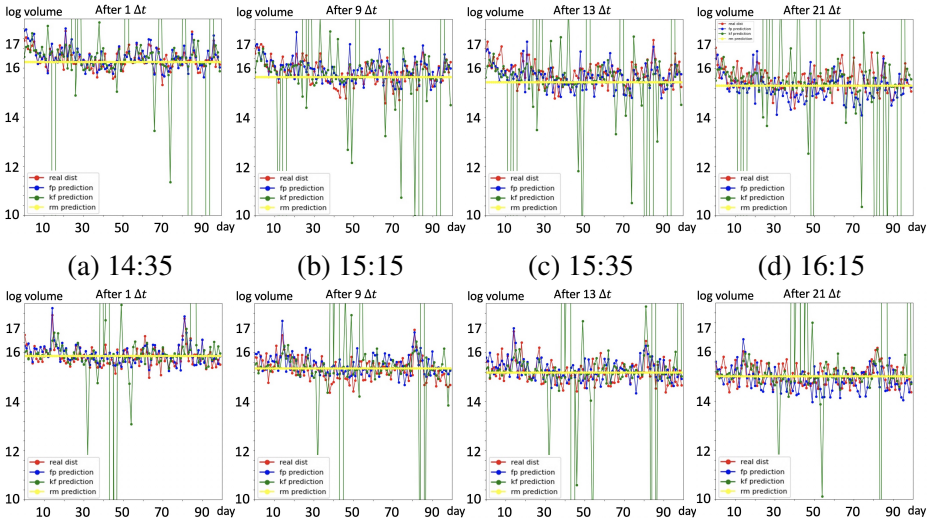
(a) D2 of Mt1 (b) D7 of Mt1 (c) D2 of Mt2 (d) D7 of Mt2

Figure 6.7: The performance comparisions among different models on D2 and D7 of Tdh and Gsn.

Table 6.2: The Wasserstein error of different models on Supplementary RNA-sequence data sets.

Data	Task	Dimension	NN	LEGEND	Ours
RNA-Tdh	D2	10	16.28	5.75	**2.15**
	D7	10	28.19	22.49	**1.03**
RNA-Gsn	D2	10	34.94	10.77	**3.31**
	D7	10	15.74	10.42	**2.07**

6.5.2 Daily Trading Volume



(a) 14:35 (b) 15:15 (c) 15:35 (d) 16:15

(e) 14:35 (f) 15:15 (g) 15:35 (h) 16:15

Figure 6.8: (a) to (d): TSLA stock. (e) to (h): GOOGL stock. We predictions of traded volume in next 100 days, RM(yellow) fails to capture the regularities of traded volume in time series, kalman filter based model(green) fails to capture noise information and make reasonable predictions, our model(blue) is able to seize the movements of traded volume and yield better predictions.

Table 6.3: The Mean absolute percentage error(MAPE) of different models on Daily Trading Volume data sets.

Stock	Time	RM	KF	Ours
JPM	14:35	0.52	0.28	**0.01**
	15:15	0.54	0.36	**0.04**
	15:35	0.51	0.42	**0.06**
	16:15	0.52	0.49	**0.12**
TSLA	14:35	0.53	0.31	**0.02**
	15:15	0.55	0.36	**0.03**
	15:35	0.53	0.39	**0.08**
	16:15	0.52	0.38	**0.14**
GOOGL	14:35	0.49	0.35	**0.01**
	15:15	0.51	0.38	**0.03**
	15:35	0.53	0.44	**0.05**
	16:15	0.51	0.42	**0.11**

6.6 Error Analysis

In this section, we provide an error analysis of our model. Suppose the hidden dynamics is driven by $g_r(\boldsymbol{x})$, the dynamics that we learn from data is $g_f(\boldsymbol{x})$, then original Itô process, Euler processes computed by true g_r and estimated g_f are

$$
\begin{aligned}
d\boldsymbol{X} &= g(\boldsymbol{X})dt + \sigma d\boldsymbol{W}, \\
\boldsymbol{x}^r_{t+\Delta t} &= \boldsymbol{x}^r_t + g_r(\boldsymbol{x}^r_t)\Delta t + \sigma\sqrt{\Delta t}\mathcal{N}(0,1), \\
\boldsymbol{x}^f_{t+\Delta t} &= \boldsymbol{x}^f_t + g_f(\boldsymbol{x}^f_t)\Delta t + \sigma\sqrt{\Delta t}\mathcal{N}(0,1),
\end{aligned}
$$

where $\boldsymbol{X}$ is the ground truth, $\boldsymbol{x}^r$ is computed by true g_r and $\boldsymbol{x}^f$ is computed by estimated g_f. Estimating the error between original Itô process and its Euler form can be very complex, hence we cite the conclusion from [140] and focus more on the error between original form and our model.

Lemma 6.6.1. *With the same initial $\boldsymbol{X}_{t_0} = \boldsymbol{x}_{t_0} = \boldsymbol{x}_0$, if there is a global Lipschitz constant*

K which satisfies

$$|g(\boldsymbol{x}, t) - g(\boldsymbol{y}, t)| \leq K|\boldsymbol{x} - \boldsymbol{y}|,$$

then after n steps, the expectation error between Itô process $\boldsymbol{x}_{t_n}$ and Euler forward process $\boldsymbol{x}^r_{t_n}$ is

$$\mathbb{E}|\boldsymbol{x}_{t_n} - \boldsymbol{x}^r_{t_n}| \leq K\left(1 + \mathbb{E}|X_0|^2\right)^{1/2} \Delta t.$$

Lemma 6.6.1 illustrates that the expectation error between original Itô process and its Euler form is not related to total steps n but time step Δt. For more details of proof process of Lemma 6.6.1, please see first two chapters in reference book [140].

Proposition 6.6.1 (Error estimation)**.** *With the same initial $\boldsymbol{x}_0$, suppose the generalization error of neural network g is ε and existence of global Lipschitz constant K*

$$|g(\boldsymbol{x}) - g(\boldsymbol{y})| \leq K|\boldsymbol{x} - \boldsymbol{y}|,$$

then after n steps with step size $\Delta t = T/n$, the expectation error between Itô process $\boldsymbol{x}_{t_n}$ and approximated forward process $\boldsymbol{x}^f_{t_n}$ is bounded by

$$\mathbb{E}|\boldsymbol{x}_{t_n} - \boldsymbol{x}^f_{t_n}| \leq \frac{\varepsilon}{K}(e^{KT} - 1) + K(1 + \mathbb{E}|\boldsymbol{x}_0|^2)^{1/2}\Delta t. \tag{6.16}$$

Proof. With initial X and first one-step iteration

$$\boldsymbol{x}_{t_0}^r = \boldsymbol{x}_{t_0}, \quad \boldsymbol{x}_{t_0}^f = \boldsymbol{x}_{t_0}, \tag{6.17}$$

$$\boldsymbol{x}_{t_1}^r = \boldsymbol{x}_{t_0}^r + g_r(\boldsymbol{x}_{t_0}^r)\Delta t + \sigma\sqrt{\Delta t}\mathcal{N}(0,1), \tag{6.18}$$

$$\boldsymbol{x}_{t_1}^f = \boldsymbol{x}_{t_0}^f + g_f(\boldsymbol{x}_{t_0}^f)\Delta t + \sigma\sqrt{\Delta t}\mathcal{N}(0,1). \tag{6.19}$$

Then we have

$$\mathbb{E}|\boldsymbol{x}_{t_0}^r - \boldsymbol{x}_{t_0}^f| = \mathbb{E}|\boldsymbol{x}_{t_0} - \boldsymbol{x}_{t_0}| = 0, \tag{6.20}$$

$$\begin{aligned}
\mathbb{E}|\boldsymbol{x}_{t_1}^r - \boldsymbol{x}_{t_1}^f| &= \mathbb{E}|\boldsymbol{x}_{t_0}^r - \boldsymbol{x}_{t_0}^f + g_r(\boldsymbol{x}_{t_0}^r)\Delta t - g_f(\boldsymbol{x}_{t_0}^f)\Delta t + \sigma\sqrt{\Delta t}\mathcal{N}(0,1) - \sigma\sqrt{\Delta t}\mathcal{N}(0,1)| \\
&\leq \mathbb{E}|\boldsymbol{x}_{t_0}^r - \boldsymbol{x}_{t_0}^f| + \mathbb{E}|g_r(\boldsymbol{x}_{t_0}^r) - g_f(\boldsymbol{x}_{t_0}^f)|\Delta t \\
&= \mathbb{E}|g_r(\boldsymbol{x}_{t_0}^r) - g_f(\boldsymbol{x}_{t_0}^r) + g_f(\boldsymbol{x}_{t_0}^r) - g_f(\boldsymbol{x}_{t_0}^f)|\Delta t \\
&\leq \mathbb{E}|g_r(\boldsymbol{x}_{t_0}^r) - g_f(\boldsymbol{x}_{t_0}^r)|\Delta t + \mathbb{E}|g_f(\boldsymbol{x}_{t_0}^r) - g_f(\boldsymbol{x}_{t_0}^f)|\Delta t \\
&\leq \varepsilon\Delta t + \mathbb{E}|g_f(\boldsymbol{x}_{t_0}^r) - g_f(\boldsymbol{x}_{t_0}^f)|\Delta t \\
&= \varepsilon\Delta t + \mathbb{E}|g_f'(\boldsymbol{x}_{t_0}^\xi)(\boldsymbol{x}_{t_0}^r - \boldsymbol{x}_{t_0}^f)|\Delta t \qquad (\boldsymbol{x}_{t_0}^\xi \in [\boldsymbol{x}_{t_0}^r, \boldsymbol{x}_{t_0}^f]) \\
&\leq \varepsilon\Delta t + K\mathbb{E}|\boldsymbol{x}_{t_0}^r - \boldsymbol{x}_{t_0}^f|\Delta t \\
&= \varepsilon\Delta t.
\end{aligned} \tag{6.21}$$

Follow the same pattern we have

$$\begin{cases}
\boldsymbol{x}_{t_2}^r = \boldsymbol{x}_{t_1}^r + g_r(\boldsymbol{x}_{t_1}^r)\Delta t + \sigma\sqrt{\Delta t}\mathcal{N}(0,1), \\
\boldsymbol{x}_{t_2}^f = \boldsymbol{x}_{t_1}^f + g_f(\boldsymbol{x}_{t_1}^f)\Delta t + \sigma\sqrt{\Delta t}\mathcal{N}(0,1),
\end{cases} \tag{6.22}$$

...

$$\begin{cases}
\boldsymbol{x}_{t_n}^r = \boldsymbol{x}_{t_{n-1}}^r + g_r(\boldsymbol{x}_{t_{n-1}}^r)\Delta t + \sigma\sqrt{\Delta t}\mathcal{N}(0,1), \\
\boldsymbol{x}_{t_n}^f = \boldsymbol{x}_{t_{n-1}}^f + g_f(\boldsymbol{x}_{t_{n-1}}^f)\Delta t + \sigma\sqrt{\Delta t}\mathcal{N}(0,1).
\end{cases} \tag{6.23}$$

Which leads to:

$$\begin{aligned}\mathbb{E}|\boldsymbol{x}^r_{t_2} - \boldsymbol{x}^f_{t_2}| &= \mathbb{E}|\boldsymbol{x}^r_{t_1} - \boldsymbol{x}^f_{t_1} + g_r(\boldsymbol{x}^r_{t_1})\Delta t - g_f(\boldsymbol{x}^f_{t_1})\Delta t + \sigma\sqrt{\Delta t}\mathcal{N}(0,1) - \sigma\sqrt{\Delta t}\mathcal{N}(0,1)| \\ &\leq \mathbb{E}|\boldsymbol{x}^r_{t_1} - \boldsymbol{x}^f_{t_1}| + \mathbb{E}|g_r(\boldsymbol{x}^r_{t_1}) - g_f(\boldsymbol{x}^f_{t_1})|\Delta t \\ &\leq \mathbb{E}|\boldsymbol{x}^r_{t_1} - \boldsymbol{x}^f_{t_1}| + \varepsilon\Delta t + K\mathbb{E}|\boldsymbol{x}^r_{t_1} - \boldsymbol{x}^f_{t_1}|\Delta t \\ &\leq (1 + K\Delta t)\varepsilon\Delta t + \varepsilon\Delta t, \end{aligned} \tag{6.24}$$

...

$$\mathbb{E}|\boldsymbol{x}^r_{t_n} - \boldsymbol{x}^f_{t_n}| \leq \varepsilon\Delta t\sum_{i=0}^{n-1}(1 + K\Delta t)^i. \tag{6.25}$$

Now let $S = \sum_{i=0}^{n-1}(1 + K\Delta t)^i$, then consider followings

$$\begin{aligned} S(K\Delta t) &= S(1 + K\Delta t) - S \\ &= \sum_{i=1}^{n}(1 + K\Delta t)^i - \sum_{i=0}^{n-1}(1 + K\Delta t)^i \\ &= (1 + K\Delta t)^n - 1 \\ &= (1 + K\frac{T}{n})^n - 1 \\ &\leq e^{KT} - 1. \end{aligned} \tag{6.26}$$

Finally we have:

$$\mathbb{E}|\boldsymbol{x}^r_{t_n} - \boldsymbol{x}^f_{t_n}| \leq \frac{\varepsilon}{K}(e^{KT} - 1), \tag{6.27}$$

$$\mathbb{E}|\boldsymbol{x}_{t_n} - \boldsymbol{x}^f_{t_n}| \leq \frac{\varepsilon}{K}(e^{KT} - 1) + K(1 + E|\boldsymbol{x}_0|^2)^{1/2}\Delta t. \tag{6.28}$$

□

Proposition 6.6.1 implies that besides time step size Δt, our expectation error interacts

with three factors, generalization error, Lipschitz constant of g and total time length. In our experiments, we find the best way to decrease the expectation error is reducing the value of K and n.

6.7 Learning Data-driven Hamiltonian System

Next we also want to apply our method on learning data-driven Hamiltonian system. A Hamiltonian system refers to a dynamical system governed by Hamilton's ordinary differential equations (ODEs). The ODEs that express Hamiltonian dynamics are famous for both their mathematical elegance and their challenges to numerical integration techniques. Symplectic integrators have been developed to make the conserved energy in Hamiltonian systems, thereby usually being more stable and structure-preserving than non-symplectic ones [141]. Popular symplectic ones include symplectic-Euler and Stömer-Verlet [141, 142], which also be our integrators in this chapter.

Hamiltonian systems exist vastly in our world, one simple case is mass-spring system and one example with more complexity is planetary movement of three bodies. The beautiful mathematical properties and behaviours Hamiltonian systems make themselves popular and a lot of researchers have built up several ways to study the systems in data-driven frameworks. Several classical works include [143, 144, 145, 146, 147, 148, 149] where researchers analyze latent differential equations in time series. Learning vector field of Hamiltonian is one of efficient ways. [150, 151, 152, 153, 154, 155, 156] learn the vector field through regression, [157, 158, 159, 160] learn the vector field via neural networks and [161] learn through Gaussian process. Some machine learning based models include [162, 163, 164, 165, 166], where popular frameworks such as RNN, LSTM, CNN and transformer are utilized.

Learning quantities that produces the Hamiltonian vector field is another way to analyze the system. Typical works include [167], where researchers proposes to assume the Hamiltonian function $H(q;p)$ as a multilayer neural network. The partial derivatives of this

network are then trained to match the time derivatives $\dot{p}$ and $\dot{q}$ observed along the trajectories in state space. [168] trains neural networks to approximate the total energy function for a pendulum. [169] learns the Hamiltonian by matching its symplectic integration with the training sequences, and its prediction is then given by the symplectic map, which is the symplectic integration of the learned Hamiltonian. [170] represents the symplectic map by a neural-network-approximated generating function. [171] constructs specialized neural networks, which represent only symplectic maps, to directly approximate the latent evolution map. Other works include [172, 173, 174, 175, 176] also cover mechanical problems modeled by Hamiltonian systems.

Most of existing data-driven works rely on the trajectory level data, where each individual's information and correspondence are known, thus L_2 based objective function is widely used. In our work, we explore the possibility to learn Hamiltonian within an aggregate data setting, where we do not assume each individual's full trajectory and correspondence. Specifically, we learn Hamiltonians by minimizing accumulating Wasserstein distance along time series observations, which is smiliar to the work of [177], where the techniques are used to infer the drift term in stochastic differential equations. Notice that [167, 169] learn the Hamiltonian that generates the vector field directly, [170] approximates generating function that corresponds to the symplectic evolution map, which can also be viewed as an "implicit learning" of the Hamiltonian. Though our method aims at learning the Hamiltonian explicitly, our problem is defined within an aggregate data setting. In conclusion, in this work we develop a method to directly study Hamiltonians in an aggregate data setting, we also show the effectiveness of our model through two experiments.

6.7.1 Hamiltonian System

Similar to the SDE system we introduced before, within settings of Hamiltonian system, we consider

$$dX = J^{-1}\nabla H_0(X)dt + \sum_{i=1}^{m} J^{-1}\nabla H_i(X) \circ dW_t^i,\ X(0) = x_0, \tag{6.29}$$

where $X \in \mathbb{R}^{2d}$, $J = \begin{bmatrix} 0 & I_{d\times d} \\ -I_{d\times d} & 0 \end{bmatrix}$, H_0 is the Hamiltonian function of the deterministic part, $H_i, i = 1, \cdots, m$, is the Hamiltonian function of the stochastic part.

Denote $X = (P, Q)$, where P represents velocity and Q represents position of individuals. Then the above system can be written as

$$\begin{aligned} dP &= -\frac{\partial H_0}{\partial Q}(P(t), Q(t))dt - \sum_{i=1}^{d} \frac{\partial H_i}{\partial Q}(P(t), Q(t)) \circ dW_t^i, \\ dQ &= \frac{\partial H_0}{\partial P}(P(t), Q(t))dt + \sum_{i=1}^{d} \frac{\partial H_i}{\partial P}(P(t), Q(t)) \circ dW_t^i. \end{aligned} \tag{6.30}$$

For instance, if we choose $H_0 = \frac{1}{2}|P|^2$ and $H_i = 0, i = 1, \cdots, m$, above system becomes

$$dP = 0,\ dQ = P.$$

6.7.2 Algorithm

Suppose individuals evolves following the H_0 and H_1, without trajectory information, we only have aggregate level data, thus we need to derive the dynamics of particles in density level.

First of all we rewrite the stochastic Hamiltonian system 6.30 in its equivalent Itô form,

i.e,

$$dX(t) = J^{-1}\nabla H_0(X(t))dt + \sum_{i=1}^{m} J^{-1}\nabla H_i(X(t))dW_t^i, \tag{6.31}$$

$$+ \frac{1}{2}\sum_{i=1}^{m} J^{-1}\nabla^2 H_i(X(t))J^{-1}\nabla H_i(X(t))dt. \tag{6.32}$$

Then the Vlasov equation of (6.29) reads

$$\begin{aligned}\frac{\partial \rho(X,t)}{\partial t} &= -\nabla\cdot(\rho(X,t)J^{-1}\nabla H_0(X)) - \frac{1}{2}\sum_{i=1}^{m}\nabla\cdot[(J^{-1}\nabla^2 H_i(X))(J^{-1}\nabla H_i(X)\rho(X,t))] \\ &\quad + \frac{1}{2}\sum_{i=1}^{m}\nabla^2\cdot[(J^{-1}\nabla H_i(X))(J^{-1}\nabla H_i(X))^\top\rho(X,t)] \\ \rho(X,0) &= p(X),\end{aligned} \tag{6.33}$$

which can be rewritten as

$$\frac{\partial \rho(X,t)}{\partial t} = -\sum_{k=1}^{2d}\frac{\partial}{\partial x_i}[(J^{-1}\nabla H_0(X))_k\rho(X,t)] \tag{6.34}$$

$$+ \frac{1}{2}\sum_{i=1}^{m}\sum_{i=1}^{2d}\frac{\partial}{\partial x_i}\Big\{(J^{-1}\nabla H_i(X))_k\sum_{j=1}^{2d}\frac{\partial}{\partial x_j}[J^{-1}\nabla H_i(X))_j\rho(X,t)]\Big\}. \tag{6.35}$$

The weak form of above density evolution is

$$\begin{aligned}&\int_X \frac{\partial\rho(X,t)}{\partial s}f(X)dX = \int_X J^{-1}\nabla H_0(X)\rho(X,t)\frac{\partial}{\partial X}f(X)dX \\ &+ \frac{1}{2}\sum_{i=1}^{m}\sum_{i=1}^{2d}\Big\{(J^{-1}\nabla H_i(X))_k\sum_{j=1}^{2d}\frac{\partial}{\partial x_j}[J^{-1}\nabla H_i(X))_j\rho(X,t)]\Big\}\frac{\partial}{\partial x_i}f(X)dX.\end{aligned} \tag{6.36}$$

In this chapter we only discuss the non-noise version of Hamiltonian system. Thus if we ignore the noise part, we simplify 6.36 as

$$\int_X \frac{\partial\rho(X,t)}{\partial s}f(X)dX = \int_X J^{-1}\nabla H_0(X)\rho(X,t)\frac{\partial}{\partial X}f(X)dX. \tag{6.37}$$

Moreover, when given true $\hat{\rho}_{t-1}$, the estimated $\tilde{\rho}_t$ is

$$\tilde{\rho}_t = \rho_{t-1} + \int_{t-1}^{t} \frac{\partial \rho}{\partial s} ds. \tag{6.38}$$

Finally the Wasserstein distance between $\hat{\rho}_t$ and $\tilde{\rho}_t$ is

$$\begin{aligned}
W(\rho_t, \tilde{\rho}_t) &= \sup_{f, ||\nabla f|| \leq 1} \mathbb{E}_{\hat{X} \sim \hat{\rho}_t}\{f(\hat{X})\} - \mathbb{E}_{\tilde{X} \sim \tilde{\rho}_t}\{f(\tilde{X})\} \\
&= \sup_{f, ||\nabla f|| \leq 1} \int_X f(X)\hat{\rho}_t dX - \int_X f(X)\tilde{\rho}_t dX \\
&= \sup_{f, ||\nabla f|| \leq 1} \int_X f(X)\hat{\rho}_t dX - \int_X f(X)(\hat{\rho}_{t-1} + \int_{t-1}^{t} \frac{\partial \rho}{\partial s} ds) dX \\
&= \sup_{f, ||\nabla f|| \leq 1} \int_X f(X)\hat{\rho}_t dX - \int_X f(X)\hat{\rho}_{t-1} dX - \int_X \int_{t-1}^{t} f(X) \frac{\partial \rho}{\partial s} ds dX \\
&= \sup_{f, ||\nabla f|| \leq 1} \int_X f(X)\hat{\rho}_t dX - \int_X f(X)\hat{\rho}_{t-1} dX \\
&\qquad - \int_{t-1}^{t} \int_X J^{-1} \nabla H_0(X) \rho(X, t) \frac{\partial}{\partial X} f(X) dX ds \\
&= \sup_{f, ||\nabla f|| \leq 1} \mathbb{E}_{\hat{X} \sim \hat{\rho}_t}\{f(X)\} - \mathbb{E}_{\hat{X} \sim \hat{\rho}_{t-1}}\{f(X)\} \\
&\qquad - \int_{t-1}^{t} \mathbb{E}_{X \sim \rho_s}\{\sum_{i=1}^{2d} J^{-1} \nabla H_0(X) \frac{\partial}{\partial X} f(X)\} ds
\end{aligned} \tag{6.39}$$

Through experiments we found that computing last term in above equation 6.39 consumes long computation time. We then propose another method to evaluate the Wasserstein distance between generated particles and observed particles as

$$\begin{aligned}
W(\hat{\rho}_t, \tilde{\rho}_t) &= \sup_{f, ||\nabla f|| \leq 1} \mathbb{E}_{\hat{X} \sim \hat{\rho}_t}\{f(\hat{X})\} - \mathbb{E}_{\tilde{X} \sim \tilde{\rho}_t}\{f(\tilde{X})\}, \\
\tilde{X}(t) &= \hat{X}(t-1) + J^{-1} \nabla H_0(\hat{X}(t-1)) \Delta t.
\end{aligned} \tag{6.40}$$

It can be proved that 6.39 and 6.40 are equivalent to each other. Till this end we con-

clude our objective function as

$$\min_{H_\omega}\left\{\sum_{n=1}^{J}\sup_{\|\nabla f_n\|\le 1}\left\{\frac{1}{N}\sum_{k=1}^{N}f_n(\tilde{x}_{t_n}^{(k)})-\frac{1}{N}\sum_{k=1}^{N}f_n(\hat{x}_{t_n}^{(k)})\right\}+\mathcal{F}\right\}, \tag{6.41a}$$

$$\mathcal{F}=\sum_{n=1}^{J}\Bigg(\left(\mathbb{E}\left[H_\omega(\tilde{x}_{t_n}^{(k)})\right]-\mathbb{E}\left[H_\omega(\hat{x}_{t_n}^{(k)})\right]\right)^2 + \left(\mathbb{E}\left[H_\omega(\tilde{x}_{t_n}^{(k)})\right]-\mathbb{E}\left[H_\omega(\hat{x}_{t_0}^{(k)})\right]\right)^2\Bigg) \tag{6.41b}$$

$$dX(t)=J^{-1}\nabla H_0(X(t))dt, \tag{6.41c}$$

$$\hat{X}(0)=X_0, \tag{6.41d}$$

where 6.41b is due to the conserved energy in the Hamiltonian system, we add this constraint to facilitate our training process. We set H_0 in 6.40 as a neural network H_ω, as well as f for the purpose of evaluating Wasserstein distance. Our algorithm can be concluded as Algorithm 7.

6.7.3 Experiments

In this section we test our algorithm on two experiments. We generate data by Symplectic Euler method [14] as

$$\begin{aligned} q(t+h)&=q(t)+h\nabla H_p(q(t),p(t)),\\ p(t+h)&=p(t)-h\nabla H_q(q(t+h),p(t)). \end{aligned} \tag{6.42}$$

As for the Hamiltonian system we are going to learn, we start with a simple quadratic system which reads

$$\begin{aligned} H_0(p,q)&=\frac{1}{2}(p_1^2+p_2^2)+\frac{1}{2}(q_1^2+q_2^2),\\ p(0)\sim\mathcal{N}(0,1),&\quad q(0)\sim\mathcal{N}(0,1). \end{aligned} \tag{6.43}$$

Algorithm 7 Learning Hamiltonian System

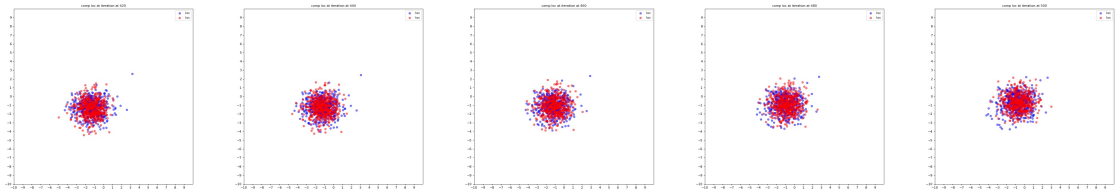
Require: Initialize f_{θ_n} $(1 \leq n \leq J)$, g_ω

Require: Set ϵ_{f_n} as the inner loop learning rate for f_{θ_n} and ϵ_g as the outer loop learning rate for H_ω

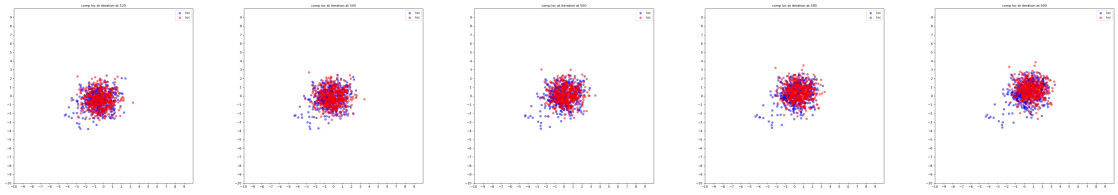
for # training iterations **do**
  for k steps **do**
    for observed time t_s in $\{t_1, ..., t_J\}$ **do**
      Compute the generated data set $\tilde{X}_{t_s}$ from 6.41c and 6.41d for $1 \leq s \leq J$
      Acquire data sets $\hat{X}_{t_s} = \{\hat{\boldsymbol{x}}_{t_s}^{(1)}, ..., \hat{\boldsymbol{x}}_{t_s}^{(N)}\}$ from real distribution $\hat{p}(\cdot, t_s)$ for $1 \leq s \leq J$
    end for
    For each dual function f_{θ_n}, compute: $\frac{1}{N}\sum_{k=1}^{N} f_{\theta_n}(\tilde{x}_{t_n}^{(k)}) - \frac{1}{N}\sum_{k=1}^{N} f_{\theta_n}(\hat{x}_{t_n}^{(k)})$
    Update each f_{θ_n} by:
    $\theta_n \leftarrow \theta_n + \epsilon_{f_n} \nabla_\theta \left(\frac{1}{N}\sum_{k=1}^{N} f_{\theta_n}(\hat{\boldsymbol{x}}_{t_n}^{(k)}) - \frac{1}{N}\sum_{k=1}^{N} f_{\theta_n}(\tilde{\boldsymbol{x}}_{t_0}^{(k)}) \right)$
  end for
  Compute $\mathcal{F} = \sum_{n=1}^{J} \Big(\big(\mathbb{E}\left[H_\omega(\tilde{x}_{t_n}^{(k)}) \right] - \mathbb{E}\left[H_\omega(\hat{x}_{t_n}^{(k)}) \right] \big)^2$
    $+ \big(\mathbb{E}\left[H_\omega(\tilde{x}_{t_n}^{(k)}) \right] - \mathbb{E}\left[H_\omega(\hat{x}_{t_0}^{(k)}) \right] \big)^2 \Big)$
  Update H_ω by:
  $\omega \leftarrow \omega - \epsilon_H \nabla_\omega \left(\sum_{n=1}^{J} \left(\frac{1}{N} f_{\theta_n}(\hat{\boldsymbol{x}}_{t_n}^{(k)}) - \frac{1}{N} f_{\theta_n}(\hat{\boldsymbol{x}}_{t_0}^{(k)}) \right) + \mathcal{F} \right)$
end for

We set H_ω as a Resnet [178] with 3 hidden layers and each layer has 36 nodes, we set the discriminator f_n as the same neural networks we defined in learning SDE earlier this chapter. The training set contains observations at 8 different time points, there will be $50\Delta t$ between each time point. The number of points at each time point we observe is 1000. We train the networks on first $400\Delta t$ interval and test the performance on second $400\Delta t$ interval. Δ is set to be 0.1. The result of q_1 versus q_2 is shown in Figure 6.9 and p_1 versus p_2 is shown in Figure 6.10. We see the particles generated (blue) by the learned H_ω almost cover all ground truth area (red).

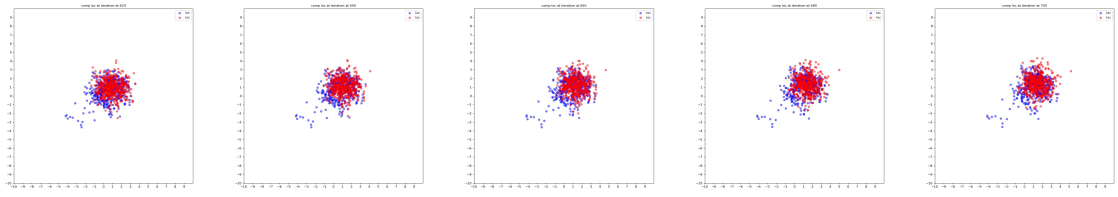
For the second experiment we study the Lotka–Volterra Model, which models the pop-



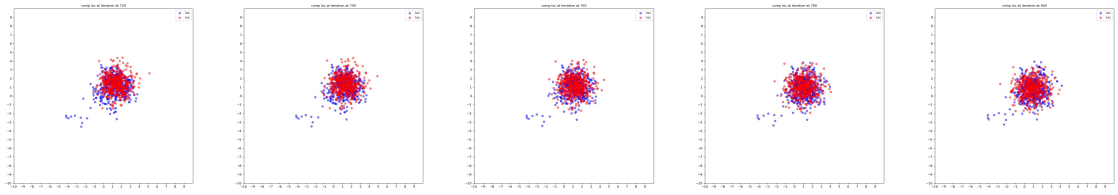
(a) $420\Delta t$ (b) $440\Delta t$ (c) $460\Delta t$ (d) $480\Delta t$ (e) $500\Delta t$



(f) $520\Delta t$ (g) $540\Delta t$ (h) $560\Delta t$ (i) $580\Delta t$ (j) $600\Delta t$



(k) $620\Delta t$ (l) $640\Delta t$ (m) $660\Delta t$ (n) $680\Delta t$ (o) $700\Delta t$



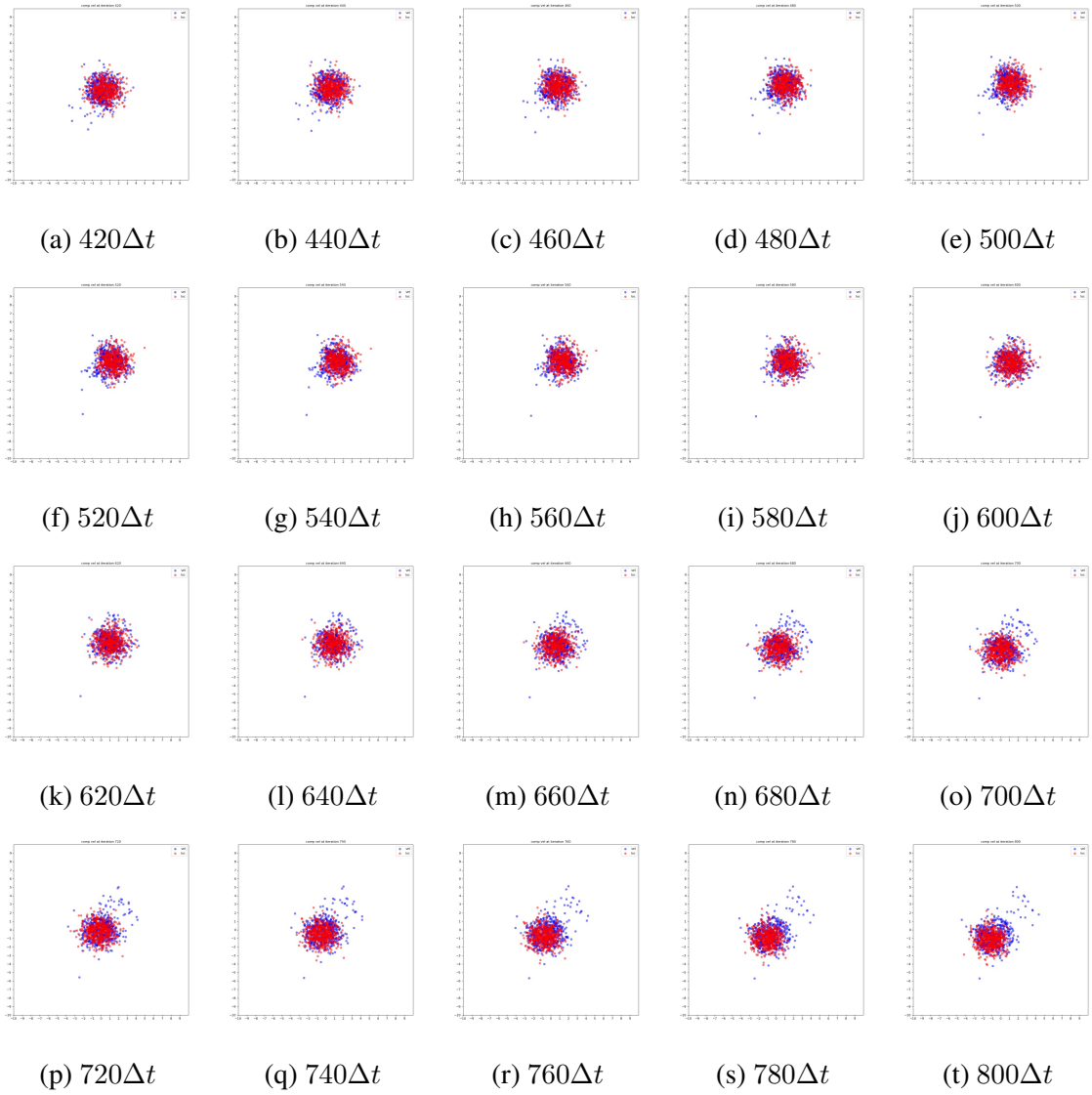
(p) $720\Delta t$ (q) $740\Delta t$ (r) $760\Delta t$ (s) $780\Delta t$ (t) $800\Delta t$

Figure 6.9: Comparison of generated q (blue) and ground truth of q (red) from $420\Delta t$ to $800\Delta t$, x-axis:q_1, y-axis:q_2.

ulations of predators and prey as they interact with each other. The system reads

$$H_0(p,q) = ap - be^p + cq - de^q,$$
$$p(0) \sim \mathcal{N}(1, 0.5), \quad q(0) \sim \mathcal{N}(1, 0.5). \tag{6.44}$$

We set H_ω and f the same structures in first experiment. The training set contains observations at 8 different time points, there will be $20\Delta t$ between each time point. The



(a) $420\Delta t$ (b) $440\Delta t$ (c) $460\Delta t$ (d) $480\Delta t$ (e) $500\Delta t$

(f) $520\Delta t$ (g) $540\Delta t$ (h) $560\Delta t$ (i) $580\Delta t$ (j) $600\Delta t$

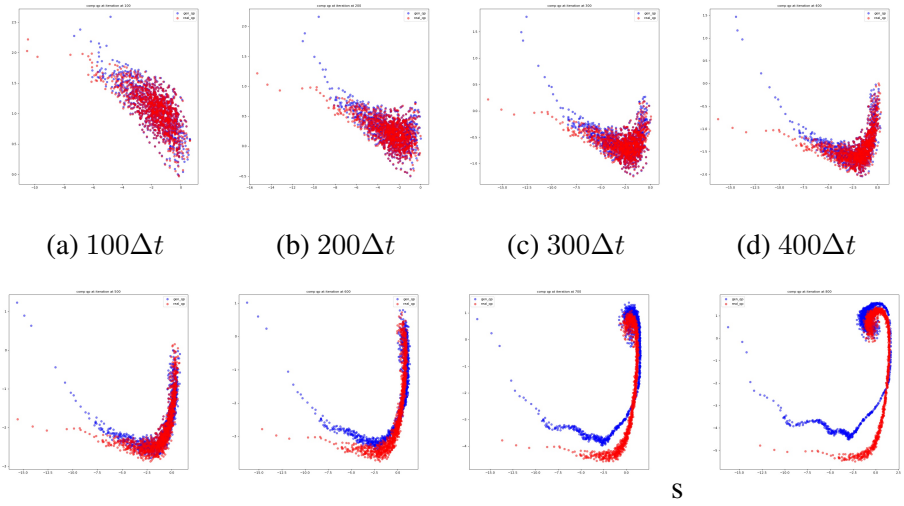
(k) $620\Delta t$ (l) $640\Delta t$ (m) $660\Delta t$ (n) $680\Delta t$ (o) $700\Delta t$

(p) $720\Delta t$ (q) $740\Delta t$ (r) $760\Delta t$ (s) $780\Delta t$ (t) $800\Delta t$

Figure 6.10: Comparison of generated p (blue) and ground truth of p (red) from $420\Delta t$ to $800\Delta t$, x-axis:p_1, y-axis:p_2.

number of points at each time point we observe is 1000. We train the networks on first $100\Delta t$ interval and test the performance on total $800\Delta t$ length. Δ is set to be 0.1. The result of q versus p is shown in Figure 6.11. We see the particles generated (blue) by the learned H_ω almost cover all ground truth area (red).

Remark 8. *We would like to mention that if the distribution of q or p doesn't change along time, then our distribution based model will not work since $W(\hat{\rho}_0, \hat{\rho}_t)$ will be very small for any time point t. Thus given $\hat{x}_0$, $\nabla H = 0$ will give small discrepancy between the*



(a) $100\Delta t$ (b) $200\Delta t$ (c) $300\Delta t$ (d) $400\Delta t$

s

(e) $500\Delta t$ (f) $600\Delta t$ (g) $700\Delta t$ (h) $800\Delta t$

Figure 6.11: Comparison of generated $q - p$ (blue) and ground truth of $q - p$ (red) from $100\Delta t$ to $800\Delta t$, x-axis:q, y-axis:p.

generated data $\tilde{x}_0$ *and the ground truth* $\hat{x}_0$.

6.8 Conclusion

In this chapter, we formulate a novel method to recover the hidden dynamics from aggregate data. In particular, our work shows one can simulate the evolving process of aggregate data as an Itô process, in order to investigate aggregate data, we derive a new model that employs the weak form of FPE as well as the framework of WGAN. Furthermore, we prove the theoretical guarantees of the error bound of our model. Finally we demonstrate our model through experiments on three synthetic data sets and two real-world data sets. As an extension of this method, we apply our model onto learning Hamiltonian system under aggregate data setting. We parametrize Hamiltonian as a neural network and developed the objective function to study the system. We also prove our method on two synthetic data sets.

CHAPTER 7

OPTIMAL DENSITY CONTROL

7.1 Introduction

Optimal control problems have been studied in a lot of areas in recent years. Typical individual-level control framework aims to control individuals precisely, the number of agents to be controlled is small. As for density control problems, the densities are usually known in advance, which can be hardly constrained in realities, especially when surrounding environment is complex. In this chapter, we provide an idea to parametrize the control as a neural network, as well as a vector field, to realize density control. Our work can be concluded as: 1) Review most recent popular optimal control frameworks. 2) Design a data-driven model to learn the optimal density control strategy $U(x)$ to drive one distribution to another distribution in sample level. 3) We also apply our model on realistic application, for example path planning of underwater glider. In an optimal control problem, we find a control $u(t)$ to steer agents from $x(0)$ to the neighbor of $x(T)$. The common setting of optimal control problem reads

$$\begin{cases} dx = f(x(t), u(t))dt, \\ x(0) = x_0, \end{cases} \tag{7.1}$$

and the reward(cost) function that needs to be maximized(minimized) is

$$L(u(\cdot)) = \int_0^T r(x(t), u(t))dt + g(x(T)), \tag{7.2}$$

where $g(x)$ is the terminal reward(cost) function.

Accordingly, if we add a Wiener process W_t, then it comes to stochastic optimal control

which reads

$$\begin{cases} dX = F(X(t), U(t))dt + \sigma dW_t, \\ X(0) = X_0, \end{cases} \tag{7.3}$$

and the reward (cost) function that needs to be maximized (minimized) is in an expectation form:

$$L(u(\cdot)) = \mathbb{E}\left\{ \int_0^T r(X(t), U(t))dt + g(X(T)) \right\}, \tag{7.4}$$

Above two basic forms are open-loop control problems, if we write control $U(t)$ as $U(x, t)$, then it comes to the feedback control problem, which falls into closed-loop control. Moreover, recent research leverages optimal transport and provides a density control framework:

$$\begin{cases} dX(t) = F(B(X(t), t), G(X(t)))dt + \sigma dW_t, \\ X(0) \sim \rho_0, \qquad X(T) \sim \rho_1, \end{cases} \tag{7.5}$$

and minimize

$$L(B(\cdot)) = \mathbb{E}\left\{ \int_0^T \frac{1}{2}||B(X(t), t)||^2 + g(X(t))\ dt \right\}, \tag{7.6}$$

Furthermore, one formulation of optimal density control [11, 12] reads

$$\begin{aligned} &\min_U L(U) = \mathbb{E}\left\{ \int_0^1 \frac{1}{2\sigma}||U(X, t)||^2 dt \right\}, \\ &\text{subject to: } dX_t = Udt + \sqrt{\sigma} dW_t, \\ &\qquad\qquad X_0 \sim \rho_0(x), \ \ X_1 \sim \rho_1(x), \end{aligned} \tag{7.7}$$

where the target is to minimize the total control energy and the particles evolve in a stochas-

tic dynamic with the control $U(x)$. If we add an extra flow field $V(x_t, t)$ to the control, the formulation comes to

$$\begin{aligned} \min_{U} L(U) &= \mathbb{E}\left\{ \int_0^1 \frac{1}{2\sigma} ||U(X,t)||^2 dt \right\}, \\ \text{subject to: } & dX_t = (V(X_t,t) + U)dt + \sqrt{\sigma} dW_t, \\ & X_0 \sim \rho_0, \;\; X_1 \sim \rho_1. \end{aligned} \tag{7.8}$$

Above problems can also be rewritten as a density based form:

$$\begin{aligned} \min_{U,\rho} L(U) &= \int_X \int_0^1 \frac{1}{2} ||U(X,t)||^2 \rho(X,t) dt dX, \\ \text{subject to: } & \frac{\partial \rho}{\partial t} + \nabla \cdot (U\rho) - \frac{\sigma}{2} \Delta \rho = 0, \\ & \rho(0,X) = \rho_0, \;\; \rho(1,X) = \rho_1. \end{aligned} \tag{7.9}$$

Generally, in the control problems we aim to minimize control energy, meanwhile the agents follow various dynamics, either in particle level or density level. The work [179] designs velocity field to make swarms density converges to the desired/commanded density distribution. [180] presents a density control by improving Smoothed Particle Hydrodynamic (SPH) [181] method with collision free condition, where one defines the density of a robot as the weighted sum of distances to its nearby robots within a certain range. [182] proposes a data-driven approach to learn the control by estimating Koopman operators. [183] provides a control strategy to steer the state from an initial distribution to a terminal one with specified mean and covariance, which is also called nonlinear covariance control. For other more general works we refer to [11, 12, 184].

We would like to point out that due to the popularity of optimal transport in recent years, several works combining Wasserstein metric and optimal control have been developed. For instance, [185] develops Wasserstein proximal algorithms to solve density control problem, where a nonlinear drift term is considered. [186] discusses optimality conditions for

optimal control problems in Wasserstein spaces. [187] studies a Wasserstein based robust control to resolve the issue that uncertain variables is unavailable. [188] proposes a novel model-predictive control (MPC) method for limiting the risk of unsafety when the distribution of the obstacles' is set to be within an ambiguity set which is measured by the Wasserstein metric. [189] sets terminal cost as Wasserstein distance and find the parameters of normalizing flow [190] by minimizing the terminal cost. [191] presents primal-dual formulation of OT control problem with existence proof and efficient numerical method, a similar method can also be found in [192] where the problem is solved by a Mean Field Game (MFG) approach. [193] calculates the individual robot trajectories by alternating two gradient flows that involve an attractive potential, a repelling function, and a process of intermittent diffusion,, in which way a large group of robots are employed to accomplish the task of shape formation.

Plenty of research of optimal control is widely applied in the domain of robotics, for a general introduction we refer to [194]. [195] presents the work of using reinforcement learning to control multi-robot system. [196, 197, 198] carefully designed an objective function to minimize the control energy, L_2 based difference between initial and target positions, meanwhile keeping collision-free conditions, notice that [198] also parametrizes the control as a neural network.

There are several common features of above works, the first point is that L_2 loss is always picked as a priority to evaluate the difference between initial and final locations. But L_2 loss is hard to be used in an aggregate setting. Second point is that even if some distribution-level metric such as Wasserstein distance is used, the target densities are already explicitly known, and Wasserstein distance is evaluated within settings of low-dimensional and small-scale of individuals. In this chapter, our goal is designing a model that can handle both high-dimensional and large-scale aggregate data in samples level by leveraging the application of neural networks.

7.2 Methodology

We parametrize the control as a neural network, when number of individuals is small, we train the control by minimizing the Chamfer distance. And when number of individuals are getting larger, we train the control by minimizing the Wasserstein distance.

If the number of agents (robots) K is moderate (no more than hundreds), then it is tractable to design control individually. For simplicity, suppose $X_i(t), U(X) \in \mathbb{R}^2$ for every time t. Let $\hat{X}_i$ be the destination locations. Then the optimal control problem is

$$\min_U L(U) := \int_0^T \underbrace{r(X(t), U(t))}_{\text{running cost}}\, dt + \underbrace{D(X(T), \hat{X})}_{\text{terminal cost}}, \tag{7.10}$$

where $X_i'(t) = U(X_i)$ for $0 \le t \le T$ and $X := (X_1; \ldots; X_K) \in \mathbb{R}^{K\times 2}$ (a matrix with K rows). Here the running cost is

$$r(X(t), U(X)) = \sum_{i=1}^{K} \sum_{j\neq i} d(\|X_i(t) - X_j(t)\|) + \sum_{i=1}^{K} \|U(X_i)\|^2, \tag{7.11}$$

for penalty function d we use

$$d(z) = (z - d_{\max}, 0)_+^2 + (d_{\min} - z, 0)_+^2, \tag{7.12}$$

to let z fall in $[d_{\min}, d_{\max}]$.

The terminal cost of measuring differences between point clouds is Chamfer distance:

$$D_{\text{Chamfer}}(X, \hat{X}) = \sum_i \min_j \|X_i - \hat{X}_j\|^2 + \sum_j \min_i \|X_i - \hat{X}_j\|^2. \tag{7.13}$$

A simple way to solve (7.10) is to discretize time, then $X_i(t+1) = X_i(t) + hU_i(X_i(t))$ for $t = 0, \ldots, T-1$ and time step size h. In this case, the unknown is $U(X_i) \in \mathbb{R}^2$ for $i = 1, \ldots, K$ and $t = 0, \ldots, T$. The gradient $\nabla_U L(U)$ can be computed, and thus we can

use gradient descent to find U, which can be easily resolved by automatic Adam method [55]. The algorithm is summarized in Algorithm 8.

Algorithm 8 Optimal Control Learning for small number of agents

Input: Samples $X(0)$ that follow initial distribution, samples $\hat{X}$ that follows target distribution, stepsize h, number of steps T, iteration numbers N_1 and N_2.
Initialize: Neural network u_θ.
for N_1 steps **do**
1. Generate full path of X until $X(T)$ by $X_i(t+1) = X_i(t) + hu_\theta(X_i(t)), 0 \le i \le K$, $0 \le t \le T-1$.
2. Compute running cost by (7.11).
3. Compute terminal cost by (7.13).
4. Update $\theta \leftarrow \theta - \tau\hat{\nabla}L(\theta)$.
end for
Output: u_θ.

If K is large (thousands or more), it is more efficient to do density control since (7.12) is hard to be computed efficiently. From a particle point of view, we need to find vector field $U(\cdot, t)$ for all t, such that $X_i'(t) = U(X_i(t), t)$ and $X_i(t)$ minimizes the cost function. The vector field $U(\cdot, t)$ can be parameterized as a DNN $U_{\alpha_t}(\cdot)$ (i.e., $U_{\alpha_t}(x) \in \mathbb{R}^2$ for each x in the domain) with time varying parameter α_t to be determined. For simplicity we set $\alpha_t = 0$. We also set $U(x) = \nabla G(x)$ for potential function $G(x)$. In this case, we reformulate our terminal cost as Wasserstein distance, and we compute Wasserstein distance by the Kantorovich-Rubinstein Duality form [6], namely, our problem is redefined as

$$r(X(t), U(X)) = \sum_{i=1}^{K} \|U(X_i)\|^2, \tag{7.14}$$

$$D_{\text{Wasserstein}}(\rho_X, \rho_{\hat{X}}) = \sup_{f, \|\nabla f\| \le 1} \left\{ \mathbb{E}_{X \in \rho_X} f(X) - \mathbb{E}_{\hat{X} \in \rho_{\hat{X}}} f(\hat{X}) \right\}. \tag{7.15}$$

We evaluate the Wasserstein distance by spectral normalization [34]. The algorithm is concluded in Algorithm 9.

Theorem 7.2.1. *Suppose the control $U(x) :\in \mathbb{R}^2 \to \mathbb{R}^2$ satisfies Lipschitz condition,*

Algorithm 9 Optimal Density Control Learning for large number of agents

Input: Samples following initial distribution $X(0)$, samples following target distribution $\hat{X}$, stepsize h, number of steps T, iteration numbers N_1 and N_2.
Initialize: Neural networks g_θ, f_η.
for N_1 steps **do**
1. Generate full path of X until $X(T)$ by $X_i(t+1) = X_i(t)+h\nabla g_\theta(X_i(t)), 0 \le i \le K$, $0 \le t \le T-1$.
2. Compute running cost by (7.14).
for N_2 steps **do**
3. Update $\eta \leftarrow \eta + \tau\hat{\nabla}\left\{\mathbb{E}_{X_T\in\rho_{X_T}} f_\eta(X_T) - \mathbb{E}_{\hat{X}\in\rho_{\hat{X}}} f_\eta(\hat{X})\right\}$.
end for
4. Update $\theta \leftarrow \theta - \tau\hat{\nabla}L(\theta)$.
end for
Output: G_θ.

namely, for $X, Y \in \mathbb{R}^2$, we have $|U(X) - U(Y)| \le C|X - Y|$, where C is Lipschitz constant. Then for $i, j \in K$, if agents i and agents j do not collide at the initial location, they will not collide in the future path which is generated by U.

Proof. At initial location, for two different locations X_0 and Y_0 we have

$$|X_0 - Y_0| \ge \sigma, \qquad |U(X_0) - U(Y_0)| \le C|X_0 - Y_0|. \tag{7.16}$$

After Δt, X_0, Y_0 are updated as

$$X_1 = X_0 + U(X_0)\Delta t, \qquad Y_1 = Y_0 + U(Y_0)\Delta t. \tag{7.17}$$

We evaluate the difference of X_1, Y_1 by

$$\begin{aligned} |X_1 - Y_1| &= |X_0 - Y_0 + (U(X_0) - U(Y_0))\Delta t| \ge |X_0 - Y_0| - |U(X_0) - U(Y_0)|\Delta t \\ &\ge |X_0 - Y_0| - C\Delta t|X_0 - Y_0| = (1 - C\Delta t)|X_0 - Y_0| \\ &= (1 - C\Delta t)\sigma. \end{aligned} \tag{7.18}$$

Follow same pattern and extend it to X_T and Y_T we have

$$|X_T - Y_T| \geq (1 - C\Delta t)^N \sigma = (1 - C\Delta t)^{\frac{1}{\Delta t}} \sigma = (1 - C\frac{T}{N})^N \sigma = e^{-CT}\sigma. \tag{7.19}$$

Thus we can tune $\Delta t, C, N, \sigma$ to bound the distance of different agents. □

Lemma 7.2.2. *Denote*

$$L(\theta) = \sup_{||\nabla f_\eta|| \leq 1} \left\{ \mathbb{E}_{X_T \in \rho_{X_T}} f_\eta(X_T) - \mathbb{E}_{\hat{X} \in \rho_{\hat{X}}} f_\eta(\hat{X}) \right\} = \sup_{||\nabla f_\eta|| \leq 1} E(\theta, f_\eta), \tag{7.20}$$

then the gradient $\nabla_\theta L(\theta)$ at any θ is given by $\nabla_\theta L(\theta) = \partial_\theta E(\theta, f_\eta(\theta))$, where $f_\eta(\theta)$ is the solution of $\sup_{||\nabla f_\eta|| \leq 1} E(\theta, f_\eta)$ for the specified θ.

Proof. From the definition we know

$$\nabla_\theta L(\theta) = \partial_\theta E(\theta, f_\eta(\theta)) + \partial_{f_\eta} E(\theta, f_\eta(\theta)) \nabla_\theta f_\eta(\theta). \tag{7.21}$$

Now we form the Lagrange function

$$\mathcal{L}(\theta, f_\eta(\theta), \mu) = E(\theta, f_\eta(\theta)) + \mu(||\nabla f_\eta|| - 1), \tag{7.22}$$

for the maximization problem $\sup_{||\nabla f_\eta|| \leq 1} E(\theta, f_\eta)$. Then the Karush-Kuhn-Tucker (KKT) condition of f_η is given by

$$\partial_{f_\eta} \mathcal{L}(\theta, f_\eta(\theta), \mu) = \partial_{f_\eta} E(\theta, f_\eta(\theta)) + \mu(\theta) ||\Delta f_\eta(\theta)|| = 0, \tag{7.23a}$$

$$\mu(\theta)(||\nabla f_\eta(\theta)|| - 1) = 0, \tag{7.23b}$$

$$\mu(\theta) \geq 0, \quad ||\nabla f_\eta(\theta)|| \leq 1. \tag{7.23c}$$

The complementary slackness condition (7.23b) implies that

$$\nabla_\theta \mu(\theta)(||\nabla f_\eta(\theta)|| - 1) + \mu(\theta)\nabla_\theta ||\nabla f_\eta(\theta)|| = 0. \tag{7.24}$$

If $\mu_(\theta) = 0$, then we know $\partial_{f_\eta} E(\theta, f_\eta(\theta)) = 0$ due to (7.23a) and hence (7.21) reduces to $\nabla_\theta L(\theta) = \partial_\theta E(\theta, f_\eta(\theta))$. If $\mu_(\theta) > 0$, then from (7.23b) we know $||\nabla f_\eta(\theta)|| - 1 = 0$, which further implies $\nabla_\theta ||\nabla f_\eta(\theta)|| = 0$ from (7.24), thus we see $\nabla_\theta L(\theta) = \partial_\theta E(\theta, f_\eta(\theta))$ from (7.21). □

Theorem 7.2.3. *Suppose the parameters θ and η are bounded in a ball centered at origin with radius R, namely, we have $\Theta := \{\theta : |\theta| \leq R\}$, $H := \{\eta : |\eta| \leq R\}$. For any $\varepsilon > 0$, let $\{\theta_j\}$ be a sequence of the network parameter in u_θ generated by the stochastic gradient descent algorithm, where $\nabla_\theta L(\theta)$ is approximated by mean of samples. If the sample complexity is $K = O(\varepsilon^{-1})$ in each iteration, then $min_{1\leq j\leq J}\mathbb{E}[|\nabla_\theta L(\theta)|^2] \leq \varepsilon$ after $J = O(\varepsilon^{-1})$ iterations.*

Proof. Since we parametrize u_θ and f_η by finite-layers neural networks, and parameters are bounded, we know u_θ and f_η have Lipschitz continuous gradient with respect to θ. Thus we say $L(\theta)$ has M-Lipschitz continuous gradient $\nabla L(\theta)$ for some $M > 0$, since $L(\theta)$ is composed of u_θ and f_η. Following the procedure of stochastic gradient descent, started from initial θ_1, the generated sequence of $\{\theta_j\}$ is

$$\theta_{j+1} = \Pi(\theta_j - \tau G_j) = \arg\min_{\theta\in\Theta}(G_j^T\theta + \frac{1}{2\tau}|\theta - \theta_j|^2), \tag{7.25}$$

where G_j denotes the stochastic gradient of $L(\theta)$ at θ_j using observed samples. Π denotes the projection of θ to the ball centered at original point with radius R and τ is the learning rate. Suppose $g_j := \nabla_\theta L(\theta_j)$ represents the true (but unknown) gradient of L at θ_j, then

we define the sequence $\{\bar{\theta}_j\}$ generated by g_j:

$$\bar{\theta}_{j+1} = \Pi(\theta_j - \tau g_j) = \arg\min_{\theta\in\Theta}(g_j^T\theta + \frac{1}{2\tau}|\theta - \theta_j|^2). \tag{7.26}$$

We would like to mention that computing $\{\bar{\theta}\}_j$ is not practical since we don't know the true g_j. To prove the convergence of the projected SGD iterations, first we utilize the property of M-Lipschitz continuity of $\nabla_\theta L(\theta)$ implies that

$$L(\theta_{j+1}) \leq L(\theta_j) + g_j^T e_j + \frac{M}{2}|e_j|^2, \tag{7.27}$$

and

$$-L(\bar{\theta}_{j+1}) \leq -L(\theta_j) - g_j^T \bar{e}_j + \frac{M}{2}|\bar{e}_j|^2, \tag{7.28}$$

where we denote $e_j := \theta_{j+1} - \theta_j$, $\bar{e}_j := \bar{\theta}_{j+1} - \theta_j$ for all i. Due to the optimality of θ_{j+1} in 7.25 we have

$$0 \leq (G_j + \frac{\theta_{j+1} - \theta_j}{\tau})^T(\bar{\theta}_{j+1} - \theta_{j+1}) = (G_j + \frac{e_j}{\tau})^T(\bar{e}_j - e_j). \tag{7.29}$$

Adding (7.27), (7.28) and (7.29) leads to

$$L(\theta_{j+1}) - L(\bar{\theta}_{j+1}) \leq (g_j - G_j)^T(e_j - \bar{e}_j) + \frac{e_j^T(\bar{e}_j - e_j)}{\tau} + \frac{M}{2}|e_j|^2 + \frac{M}{2}|\bar{e}_j|^2. \tag{7.30}$$

Repeating (7.27) to (7.30) with θ_{j+1} and $\bar{\theta}_{j+1}$ replaced by $\bar{\theta}_{j+1}$ and θ_j respectively, with the optimality of θ_{j+1} in (7.26) with g_j we have

$$L(\bar{\theta}_{j+1}) - L(\theta_j) \leq -(\frac{1}{\tau} - \frac{M}{2})|\bar{e}_j|^2. \tag{7.31}$$

We sum up (7.30) and (7.31) yielding

$$L(\theta_{j+1}) - L(\theta_j) \leq (g_j - G_j)^T(e_j - \bar{e}_j) + \frac{e_j^T(\bar{e}_j - e_j)}{\tau} + \frac{M}{2}|e_j|^2 - (\frac{1}{\tau} - M)|\bar{e}_j|^2. \tag{7.32}$$

Now due to Cauchy-Schwarz inequality, the definitions of θ_{j+1} and $\bar{\theta}_{j+1}$, we show that

$$\begin{aligned}(g_j - G_j)^T(e_j - \bar{e}_j) = (g_j - G_j)^T(\theta_{j+1} - \bar{\theta}_{j+1}) &= (g_j - G_j)^T(\Pi(\theta_j - \tau G_j) - \Pi(\theta_j) - \tau g_j) \\ &\leq |g_j - G_j||\Pi(\theta_j - \tau G_j) - \Pi(\theta_j) - \tau g_j| \leq \tau|g_j - G_j|^2, \end{aligned} \tag{7.33}$$

where the last inequality is due to the fact that the projection Π onto the convex set Θ is a non-expansive operator. Moreover we notice that

$$\frac{e_j^T(\bar{e}_j - e_j)}{\tau} = \frac{1}{2\tau}(|\bar{e}_j|^2 - |e_j|^2 - |e_j - \bar{e}_j|^2). \tag{7.34}$$

Substituting (7.33) and (7.34) into (7.32) we have

$$L(\theta_{j+1}) - L(\theta_j) \leq \tau|g_j - G_j|^2 - (\frac{1}{2\tau} - M)|\bar{e}_j|^2 - (\frac{1}{2\tau} - \frac{M}{2})|e_j|^2 - \frac{1}{2\tau}|e_j - \bar{e}_j|^2. \tag{7.35}$$

If we assume the neural network training provides us a reasonable accuracy on G_j, namely we have

$$\mathbb{E}[|G_j - g_j|^2] \leq \varepsilon. \tag{7.36}$$

Taking expectation on both sides of (7.35) and reordering leads to

$$(\frac{1}{2\tau} - M)\mathbb{E}[|\bar{e}_j|^2] \leq \mathbb{E}[L(\theta_j)] - \mathbb{E}[L(\theta_{j+1})] + \tau\varepsilon - ((\frac{1}{2\tau} - \frac{M}{2})\mathbb{E}[|e_j|^2], \tag{7.37}$$

Finally we take sum of (7.37) for $j = 1, 2, ..., J$, then divide both sides by $(\frac{1}{2\tau} - M)J$, set $\tau = \frac{1}{4M}$ we have

$$\frac{1}{J}\sum_{j=1}^{J}\mathbb{E}[|\bar{e}_j|^2] \leq \frac{16M(\mathbb{E}[\theta_1] - \mathbb{E}[\theta_{J+1}])}{J} + 4\varepsilon \leq \frac{16M(\mathbb{E}[\theta_1] - \mathbb{E}[\theta_*])}{J} + 4\varepsilon \leq \varepsilon, \quad (7.38)$$

if we set $\mathbb{E}[\theta_*]) = \min_{\theta\in\Theta}\mathbb{E}[\theta]) \geq 0$ and $J = 4\varepsilon^{-1}(16M(\mathbb{E}[\theta_1] - \mathbb{E}[\theta_{J^*}])$. □

7.3 Experiments

In this section we validate our algorithms on both synthetic and realistic data sets.

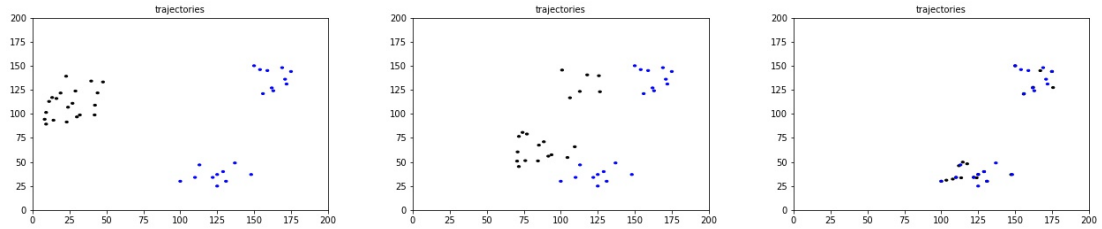
7.3.1 Synthetic Data

We first evaluate our model on several synthetic data sets which are generated in two cases: **Synthetic-1** and **Synthetic-2**.

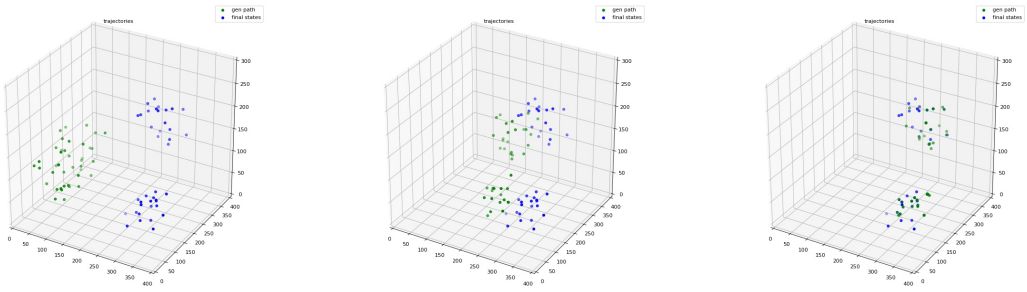
Experiment Setup: In all synthetic data experiments, we set the control $U(x)$ as a neural network with different structures, which will be stated later. We set the discriminator f as a fully-connected network, the number of hidden layers and nodes in each layer depend on the dimension of problems. The only one activation function we choose is Tanh. In terms of training process, we use the Adam optimizer [55] with learning rate 10^{-4}. Furthermore, we use spectral normalization to realize $\|\nabla f\| \leq 1$[34]. We initialize the weights with Xavier initialization[99] and train our model by Algorithm 8 and Algorithm 9 for small number of agents and large number of agent, respectively. We set the data size at each time point is $N = 2000$, treat 1500 data points as the training set and the other 500 data points as the test set, Δt and T are set to be 0.1 and 1 respectively.

Synthetic-1: In Synthetic-1, we start with Algorithm 8 on twenty two-dimensional and three-dimensional samples. Here we set our control U as the gradient of a fully connected neural network, namely, $U(x) = \nabla G(x)$, where $G(x)$ has three hidden layers and each layer has 18 nodes. Other experimental settings stay the same as introduced

before. As shown in Figure 7.1, the initial positions of samples are around left upper corner (black/green) while the target final positions are on right side with both upper and lower corners (blue). As we see, the samples gradually move to the target positions under $U(x)$. We also notice that even we choose Chamfer distance as our objective function, the initial points are not being moved to fit the target points perfectly, but close enough to satisfy the similarity on distribution level.



(a) Syn-1-2dim:t=0 (b) Syn-1-2dim:t=0.5 (c) Syn-1-2dim:t=1

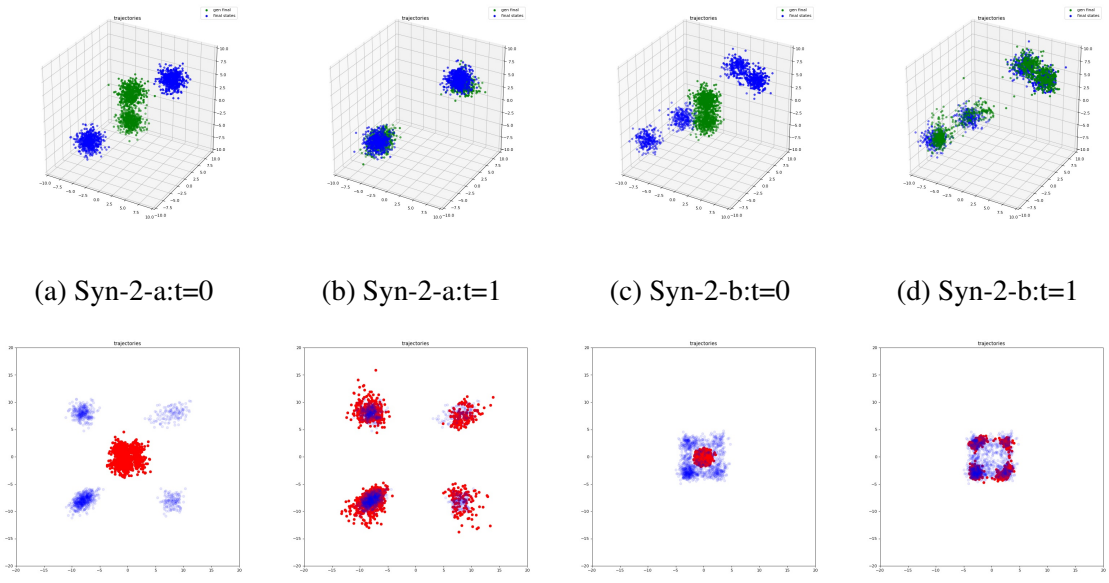


(d) Syn-1-3dim:t=0 (e) Syn-1-3dim:t=0.5 (f) Syn-1-3dim:t=1

Figure 7.1: Trajectories of samples (black/green) at different time points as they are heading to the target positions (blue) under the learned control strategy $U(x) = \nabla G(x)$, where $G(x)$ is a neural network.

Synthetic-2: In Synthetic-2, we try to control large number of agents in various scenarios. In the first group, the initial(green)/target(blue) positions are shown in Figure 7.2, the first row shows the projections of 100 dimensional cases onto 3 dimensional space. All samples follow Gaussian distributions. We see green points are moved to destinations that cover the targeting area. Here we set U as the gradient of a fully connected neural network G, where G has three hidden layers and each layer has 36 nodes. We set f as a six-hidden-

layer neural network, each layer has 256 nodes. In the second group, we also show the 2D projections of 100D cases, as shown in second row of Figure 7.2. The samples here are following unbalanced Gaussian distributions, we see that the samples at initial positions (red) finally converge to the target positions (blue) as we expect. Notice that in second group we keep the same structure of f but in Syn-2-2c we set U as Resnet [178] with 4 layers and each layer has 32 nodes, and in Syn-2-2d we set U as normalizing flow [127] with flow length 30.



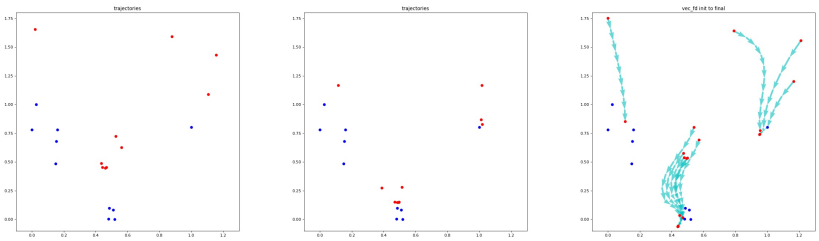
(a) Syn-2-a:t=0 (b) Syn-2-a:t=1 (c) Syn-2-b:t=0 (d) Syn-2-b:t=1

(e) Syn-2-2c:t=0 (f) Syn-2-2c:t=1 (g) Syn-2-2d:t=0 (h) Syn-2-2d:t=1

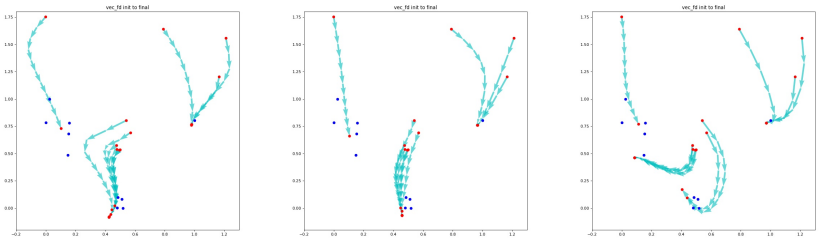
Figure 7.2: Trajectories of samples (green/red) at start and final locations, compared with target positions (blue), $U(x)$ is parametrized as Resnet and normalizing flow in Syn-2-2c and Syn-2-2d respectively.

7.3.2 Realistic Data

In this experiment we aim to control the density of Autonomous Underwater Vehicles (AUVs). AUVs are a class of submerged marine robots capable of performing persistent missions in the ocean, when controlling AUVs from initial positions to target positions, we

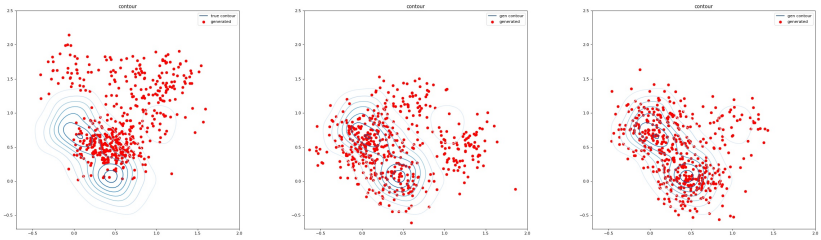


(a) Real-a:t=0 (b) Real-a:t=0.5 (c) Real-a:t=1 with traj



(d) Real-b:t=0 (e) Real-b:t=0.5 (f) Real-b:t=1

Figure 7.3: (a) to (c): Trajectories of samples (red) at different time points as they are heading to the target positions (blue) under the learned control strategy $U(x)$, (d) to (f): other trajectories that fall into local minimum. $U(x)$ is parametrized as Resnet in all cases.



(a) Real-b:t=0 (b) Real-b:t=0.5 (c) Real-b:t=1

Figure 7.4: Trajectories of samples (red) at different time points as they are heading to the target density (blue contour) under the learned control strategy $U(x)$, which is also parametrized as Resnet.

also need to consider the flow field in the ocean. Our control problem is revised as

$$\min_{U(X)} D_{\text{Wasserstein}}(\rho_X, \rho_{\hat{X}}) + \int_0^1 \int_\Omega |U(X) - V(X)|^2 \rho(X,t) dX dt,$$
$$\text{subject to: } \frac{dX}{dt} = U(X), \tag{7.39}$$

where $V(x)$ is the underwater flow field, the data of which is provided by GTSR Lab [199]. In (7.39) we aim to move AUVs to the surroundings of target locations on distribution level. We apply both algorithms 8 and 9 to realize our control strategy. Notice that the size of raw data is small (10 samples only), when we apply Algorithms 9, we augment data to the size of $N = 500$ by generating Gaussian samples around original data, namely, the final training data is following Gaussian mixtures.

The results for small size data is shown in Figure 7.3. We see that with the influence of our added flow field, the learned vector field still pushforwards the agents to the surrounding areas of our target positions. We also show the trajectories of the agents in (c), we see that the agents cannot match on one to one level, this is due to high non-convexity of the Chamfer distance, during the training process it easily falls into local minimum. We also show some results of falling into local minimum in (d), (e) and (f). However to achieve perfect matching of the initial and target positions or to get the global minimizer of Chamfer distance are difficult problems that we plan to tackle in our future research. This issue seems to be solvable when we allow U to be time dependent.

We show the result for large size data in Figure 7.4, the contour of target density is indicated by blue and we present generated points (red) from the initial distribution to the end distribution. We see that the whole distribution is moving to the target, namely, the generated points gradually fall into the range of target contour map.

7.4 Conclusion

In this chapter we develop algorithms to realize density control for both small number and large number of agents. Unlike traditional control methods, our models are able to handle high dimensional cases by leveraging neural networks. We realize density control by minimizing Wasserstein distance between generated distribution and target distribution, which is seldom used in other works as well. Our models are validated by both synthetic and realistic experiments.

CHAPTER 8

STUDIES OF TRADING STRATEGIES

8.1 Predicting Daily Trading Volume

8.1.1 Introduction

Trading volume is the total quantity of shares or contracts traded for specified securities such as stocks, bonds, options contracts, futures contracts and all types of commodities. The trading volume of any type of security can be measured during a trading day or a specified time period. In our case, daily trading volumes of stocks are measured. The trading volume is an essential component in trading alpha research since it tells investors about the market's activity and liquidity. Over the past decade, along with the improved accessibility of ultra-high-frequency financial data, evolving data-based computational technologies have attracted many attentions in financial industries [200, 201, 202, 203, 204, 205, 206, 207, 208]. Meanwhile, the development of algorithmic and electronic trading has shown great potential of trading volume since many trading models require the forecast of intraday volume as an key input [209, 210, 211]. As a result, there are growing interests in developing models for precisely predicting intraday trading volume.

Researchers aim to propose various strategies to accomplish trading efficiently in the electronic financial markets, meanwhile they wish to minimize transaction costs and market impacts [212]. The study of trading volume generally falls into two lines to achieve the goals. One line of work is focusing on suggesting optimal trading sequence and amount [213, 214, 215, 216, 217], while another line is investigating the relationships among trading volume and other financial variables or market activities such as bid-ask spread, return volatility and liquidity [218, 219, 220, 221, 222, 223, 224, 225, 226, 227], etc. Thus a precise model that provides insights of trading volume can be regarded as a basis for both

lines of work.

There are several existing methods to estimate future trading volume. As a fundamental approach, rolling means (RM) predicts intraday volume during a time interval by averaging volume traded within the same interval over the past days. The concept of RM model is straightforward, but it fails to adequately capture the intraday regularities. One classical intraday volume prediction model decomposes trading volume into three components, namely, a daily average component, an intraday periodic component, and an intraday dynamic component, then the model adopts the Component Multiplicative Error Model (CMEM) to estimate the three terms [228]. Though this model outperforms RM, the limitations such as high sensitivity to noise and initial parameters complicate its practical implementation. [137] proposes a new model to deal with the logarithm of intraday volume, which simplifies the multiplicative model into an additive one. The model is constructed within the scope of a two-state (intraday and overday features) Kalman Filter [229] framework, the authors utilize the expectation-maximization (EM) algorithm for the parameter estimation. Though the model provides a novel view to study intraday and overday factors, the flexibility is not satisfied since the model treats the number of hidden states of all stocks as two, thus there may be information loss. Moreover, from experiment we see that the dominant term in the model is actually daily seasonality, and the learning process of parameters is not robust.

As an extension of two-state Kalman Fiter, our new model has advantages such as higher prediction precision, stability and simple computation structure. In general our contributions are:

- Firstly, we develop a new method that combines cubic spline and statistical process to determine the best degrees of freedom (DOFs) for different stocks.
- Secondly, by choosing suitable DOFs, we provide a smoothing prediction of daily trading volume.

- Finally, we demonstrate that our model outperforms RM and two-state Kalman Fiter through experiments on 978 stocks.

8.1.2 Methodologies

We denote the i-th observation on day t as $vol_{t,i}$, the local indices $i \in \{0, 1, 2, ..., I\}$ and $t \in \{0, 1, 2, ..., T\}$, we set global index τ=$t * I + i$ thus $vol_{t,i}$ = vol_{τ}.

8.1.2 Two-state Kalman Filter Model

Before introducing our method, we would like to review the two-state Kalman Fiter model. Within the model, the volume is defined as the number of shares traded normalized by daily outstanding shares:

$$vol_{t,i} = \frac{\text{shares traded}_{t,i}}{\text{daily outstanding shares}_{t}}. \tag{8.1}$$

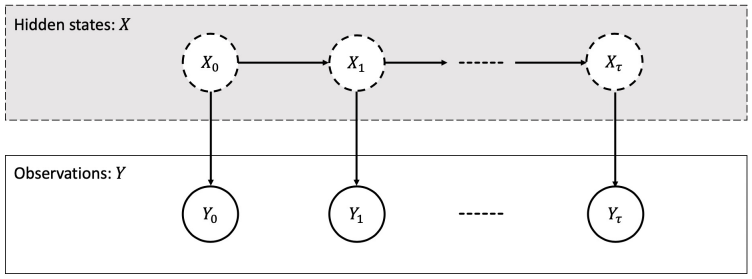


Figure 8.1: A graphical representation of the Kalman Filter model: each vertical slice represents a time instance; the top node X in each slice is the hidden state variable corresponding to the underlying volume components; and the bottom node Y in each slice is the observed volume in the market

This ratio is one way of normalization [137], the normalization is beneficial to correct low-frequency variation, which is caused by the change of trading volume. Log-volume refers to the natural logarithm of trading volume. The researchers train their model with log-volume and evaluate the predictive performance based on volume. The reason for using log-volume is that it converts the multiplicative terms [228] to additive relationships, and

makes it naturally fits Kalman Filter framework. Moreover, the logarithmic transformation facilitates to reduce the skewness property of volume data [230]. [137]'s model is built upon Kalman Fiter framework, as illustrated in Figure 8.1. X represents hidden state that is not observable, Y represents logarithm of observed traded volume. The mathematical updates are:

$$\begin{aligned} X_{\tau+1} &= A_\tau X_\tau + q_\tau, \\ Y_\tau &= Cx_\tau + \phi_\tau + v_\tau, \end{aligned} \tag{8.2}$$

for $\tau = 1, 2, ..., T * I$, where $X_\tau = [\eta_\tau \ \mu_\tau]^T$ is the hidden state vector containing two parameters, namely, the daily average part and the intraday dynamic part; $A_\tau = diag(a_\tau^\eta, a_\tau^\mu)$ is the state transition matrix; observation matrix C is fixed as $[1 \ 1]$; $q_\tau = [q_\tau^\eta \ a_\tau^\mu]^T \sim N(0, Q_\tau)$ where $Q_\tau = diag((\sigma_\tau^\eta)^2, (\sigma_\tau^\mu)^2)$; $v_\tau \sim N(0, r)$, and $\phi = [\phi_1, \phi_2, ..., \phi_I]$ is treated as the seasonality; initial state $X_1 \sim N(\pi_1, \Sigma_1)$. The unknown system parameters $\theta(\pi_1, \Sigma_1, a^\eta, a^\mu, \sigma^\eta, \sigma^\mu, r, \phi)$ are estimated by explicit equations, which are derived from expectation-maximization(EM) algorithm. For more details of two-state model, we suggest readers review the original paper [137].

8.1.2 Our Model: Various-states Kalman Filter

In the two-state model mentioned above, the DOF of hidden state variable is two since it has two factors, representing intraday and overday. Since there is no systematic way to determine a correct DOF of hidden state variable, especially for various stocks. Our concern is that how to find a better DOF for each stock and predict trading volume more precisely. As a conclusion our new method still falls into the Kalman Fiter framework,

however, we change equation (8.2) to the most common Kalman Fiter update:

$$X_{t+1} = BX_t + \gamma,$$
$$Y_t = DX_t + \psi. \tag{8.3}$$

The differences of Equation (8.2) and Equation (8.3) are as follows: X_t represents the hidden state whose dimension is $n \times 1$, where n, as the DOF of hidden state variable, will be determined in Algorithm 10; state transition matrix B is a $n \times n$ matrix while observation matrix D is a $I \times n$ matrix; $\gamma \sim N(0, \Gamma)$ where transition covariance matrix Γ is a $n \times n$ matrix; $\psi \sim N(0, \Psi)$ where observation covariance matrix Ψ is a $I \times I$ matrix; initial state $X_1 \sim N(\pi_1, \Sigma_1)$ and observation Y_t is a $I \times 1$ vector. Notice that B, D, Γ and Ψ are uniquely determined by training data.

For our model, the data we use is three years historical daily trading volume from Bloomberg. We define observation as multiplication of traded volume and Volume Weighted Average Price (VWAP).

$$vol_{t,i}^{\text{new}} = \text{Volume} \times \text{VWAP}. \tag{8.4}$$

Furthermore we model and evaluate performance by the percentage ratio:

$$p_{t,i} = \frac{vol_{t,i}^{\text{new}}}{\sum_i^I vol_{t,i}^{\text{new}}} \times 100. \tag{8.5}$$

8.1.2 DOF of State Space

Our assumption is that different stocks will have distinct number of parameters in hidden state X_τ. For a specific stock, we call the dimension of X_τ as degrees of freedom(DOFs), thus for stock with index s, $X_\tau^s = [x_1^s, x_2^s, ..., x_{dof}^s]$. One of key concerns of our model is how to determine DOFs for each stock. By experiment, we find that seasonality ϕ domi-

Algorithm 10 Find DOF by Cross Validation

Require: Training data Y, shuffle times N_s and cross validation times N_{cv}

for DOF = 1,2,...,I **do**
  for i = 1,2,...,N_s **do**
    Shuffle Y
    for j = 1,2,...,N_{cv} **do**
      Shuffle and split Y into training set and test set
      Compute cubic smoothing spline on training set
      Compute and store MSE on test set
      Compute and store standard error(SE) of each MSE
    end for
    Find best DOF by "one standard error rule"
  end for
end for

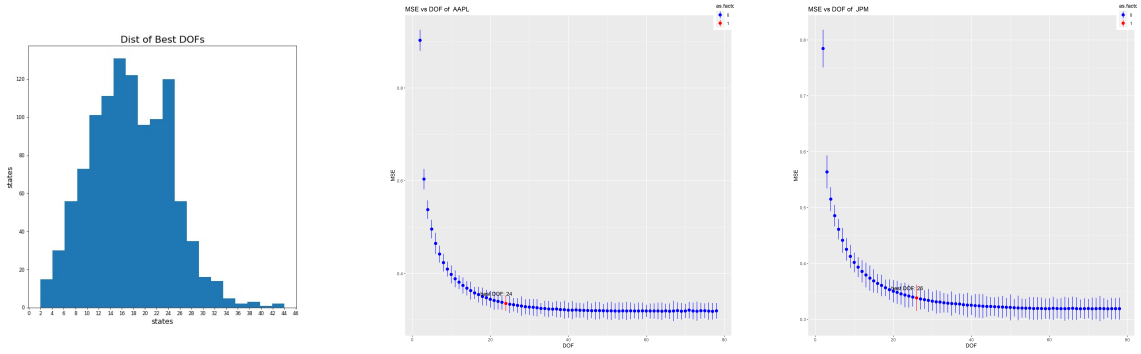
nates the prediction of traded volume in two-state model, and in both of two-state model and our model, I=77, which means in each day for each stock we have 78 observations and ϕ has 78 parameters to be determined. We aim to include seasonality in the hidden states and drop ϕ to avoid dominant term. In terms of avoiding over-fitting and reducing computation load, we use cubic spline to fit 78 observations smoothly in each day. Given a series of observations $y_i, i \in 1, 2, ..., I$, the cubic spline is to find a function g which minimizes

$$\sum_{i=1}^{I}(y_i - g(x_i))^2 + \lambda \int g''(x)^2 dx, \tag{8.6}$$

where $i = 1, 2, ..., I$ in our case, λ is a non-negative tuning parameter. The function g that minimizes 8.6 is known as a smoothing spline. The term $\sum_{i=1}^{I}(y_i - g(x_i))^2$ encourages g to fit the data well, and the term $\lambda \int g''(x)^2 dx$ is regarded as a penalty that controls smoothness of the spline. By solving 8.6 we have

$$\boldsymbol{g_\lambda} = \boldsymbol{S_\lambda y}, \tag{8.7}$$

where $\boldsymbol{g_\lambda}$, as the solution to 8.6 for a particular choice of λ, is a I-dimensional vector containing the fitted values of the smoothing spline at the training points $x_1, x_2, ..., x_I$.



(a) Distribution of best DOFs (b) Best DOF of AAPL (c) Best DOF of JPM

Figure 8.2: DOFs of states

Equation 8.7 indicates that the vector of fitted values can be written as a $I \times I$ matrix S_λ times the corresponding vector y. Then the DOF is defined to be the trace of the matrix S_λ.

Our first target is to give a specific DOF and get the corresponding spline. Thanks to the work of B. D. Ripley and Martin Maechler (https://rdrr.io/r/stats/smooth.spline.html), we are able to get fitting splines when given reasonable DOFs. After fitting process then we use cross validation to find the best DOF that achieves lowest mean squared error(MSE). Algorithm 10 outlines the mechanism of finding DOF for each stock. We analyze the DOFs of 978 stocks, the distribution of stocks' DOFs and best DOFs of some stocks are shown in Figure 8.2.

8.1.2 Kalman Filter

Once given the best DOF, we utilize Kalman Filter to do predictions. Kalman Filter is an online algorithm which precisely estimate the mean and covariance matrices of hidden states. Suppose parameters in Equation (8.3) are known, Algorithm 11 outlines the mechanism of the Kalman Filter. We model the distribution of hidden state X_{t+1} conditional on all of the percentage observations up to time t. Since we suppose γ and ψ in Equation (8.3) are Gaussian noise, thus all hidden states follow Gaussian distributions and it is only necessary to characterize the conditional mean and the conditional covariance as shown in line 3 and line 4. Then given new observations we correct the mean and covariance in line

7 and line 8.

Algorithm 11 Kalman Filtering

Require: Parameters $B, D, \Gamma, \Psi, \pi_0, \Sigma_0$

initialize $X_0 \sim N(\pi_0, \Sigma_0)$
for t=0,1,2,...,T-1 **do**
    predict next state mean: $X_{t+1|t} = BX_{t|t}$
    predict next state covariance: $S_{t+1|t} = B_t S_{t|t} B_t \top + \Gamma_t$
    obtain measurement Y_t
    compute Kalman gain: $K_{t+1} = S_{t+1|t} D^\top (D S_{t+1|t} D^\top + \Psi)^{-1}$
    update current state mean: $X_{t+1|t+1} = X_{t+1|t} + K_{t+1}(Y_t - DX_{t+1|t})$
    update current state covariance: $S_{t+1|t+1} = (I - K_{t+1}D)S_{t+1|t}$
end for

Algorithm 12 Kalman Smoothing

Require: Parameters $B, D, \Gamma, \Psi, \pi_1, \Sigma_1$ and $X_{T|T}, S_{T|T}$ from Kalman Filtering

for t=T-1,T-2,...,0 **do**
    compute: $J_t = S_{t|t} B_t^T S_{t+1|t}^{-1}$
    compute mean: $X_{t|T} = X_{t|t} + J_t(X_{t+1|T} - BX_{t+1|t})$
    compute covariance:$S_{t|T} = S_{t|t} + J_t(S_{t+1|T} - S_{t+1|t})J_t^\top$
    compute joint covariance:$S_{t,t-1|T} = S_{t|t} J_{t-1}^\top + J_t(S_{t+1,t|T} - BS_{t|t})J_{t-1}^\top$
end for
specially, $S_{T,T-1|T} = (I - K_T D)BS_{T-1|T-1}$

Our ultimate goal is to make predictions of X_t and Y_t by Equation (8.3). The method to calibrate parameters is expectation-maximization(EM) algorithm. Smoothing process infer past states before X_T conditional on all the observations from the training set, which is a necessary step in model calibration because it provides more accurate information of the unobservable states. We outlines Kalman smoothing process as Algorithm 12.

After performing Algorithm 10, 11 and 12, we need to estimate parameters by EM method, as shown in algorithm 13. EM algorithm is one common way to estimate parameters of Kalman Filter problem. It extends the maximum likelihood estimation to cases where hidden states are involved [231]. The EM iteration alternates between performing an E-step (i.e., Expectation step), which constructs a global convex lower bound of the expectation of log-likelihood using the current estimation of parameters, and an M-step (i.e.,

Maximization step), which computes parameters to maximize the lower bound found in E-step. Two advantages of EM algorithm are fast convergence and existence of closed-form solution. The derivations of Kalman Filter and EM algorithm beyond the scope of this chapter, we refer interested readers to [229] for more details.

Algorithm 13 EM algorithm

Require: Initial $B, D, \Gamma, \Psi, \pi_1, \Sigma_1$, results from Kalman Filtering and Kalman smoothing

while not converge **do**
  for t=T-1,T-2,...,0 **do**
    $P_{t+1} = S_{t+1|T} + X_{t+1|T} X_{t+1|T}^\top$
    $P_{t+1,t} = S_{t+1,t|T} + X_{t+1|T} X_{t|T}^\top$
  end for
  $\pi_1 = X_{1|T}$
  $\Sigma_1 = P_1 - X_{1|T} X_{1|T}^\top$
  $B = (\sum_{t=0}^{T-1} P_{t+1,t})(\sum_{t=2}^{T} P_t)^{-1}$
  $\Gamma = \frac{1}{T} \sum_{t=0}^{T-1} (P_{t+1} + P_t - P_{t+1,t} B^\top - B P_{t+1,t}^\top)$
  $\Psi = \frac{1}{T+1} \sum_{t=0}^{T} (Y_t Y_t^\top + D P_t D^\top - Y_t X_{t|T}^\top D^\top - D X_{t|T} Y_t^\top)$
end while

8.1.3 Experiment

8.1.3 Data Introduction

We collect and analyze intraday trading volume of 978 stocks on major U.S. markets. For example, a glance of the information of stock "AAPL" is summarized in Table 8.1. The data covers the period from January 3rd 2017 to May 29th 2020, excluding none trading days. Each trading day consists of 78 5-minute bins. Volume and percentage are computed by Equation 8.4 and 8.5 respectively. All historical data used in the experiments are obtained from the Invesco Inc.

8.1.3 Experiment Set-up

We choose two-state model and RM model mentioned before as baselines. Data from January 3rd 2017 to June 30th 2017 is considered as training set while data from July 5th

Table 8.1: Historical intrady trading volume of stock "AAPL"

Time	LAST	FIRST	HIGH	LOW	VOLUME	VWAP	time bin
1/3/2017	106.24	105.9	106.42	105.59	1139228	106.11	14:30
1/3/2017	105.28	106.24	106.34	105.19	1245847	105.77	14:35
1/3/2017	105.51	105.29	105.53	104.85	1289865	105.23	14:40
...	...	...	...	...	...	...	...
1/3/2017	106.23	106.04	106.23	106	1675070	106.09	20:55
...	...	...	...	...	...	...	...
5/29/2020	318.25	319.25	320	318.22	747433	319.21	13:30
...	...	...	...	...	...	...	...
5/29/2020	317.92	319.29	319.62	317.46	1969841	318.92	19:55

2017 to January 2nd 2018 is treated as test set. Both of training set and test set contain T_{train}=125 trading days(from day 0 to day 124). We initialize B, Γ, Ψ as identity matrices, and D as smooth matrix, then perform Algorithms 10 to 13 on traning set to obtain model parameters, finally make predictions on next T_{test}=125 days(from day 125 to day 249). We evaluate performances of three models by mean absolute percentage error(MAPE):

$$\text{MAPE} = \frac{1}{M}\sum_{\tau=1}^{M} |p_{\tau}^{\text{true}} - p_{\tau}^{\text{predicted}}|, \tag{8.8}$$

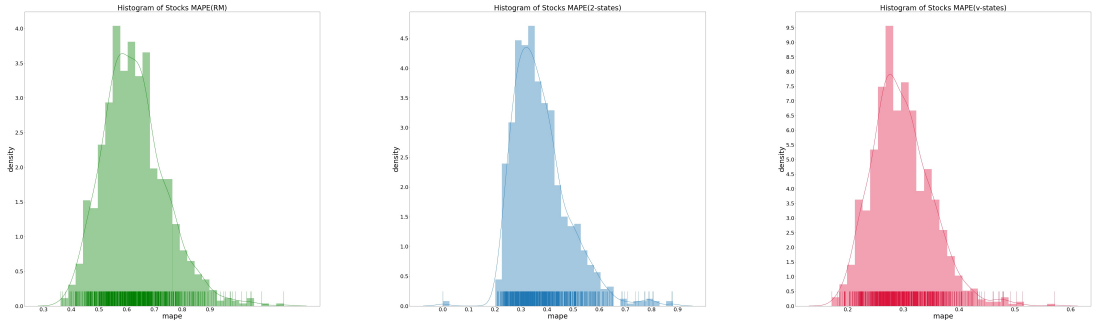
where $\tau = t * I + i$.

8.1.4 Results

In this section we compare our model with two state-of-the-art baselines and perform some analysis of our results.

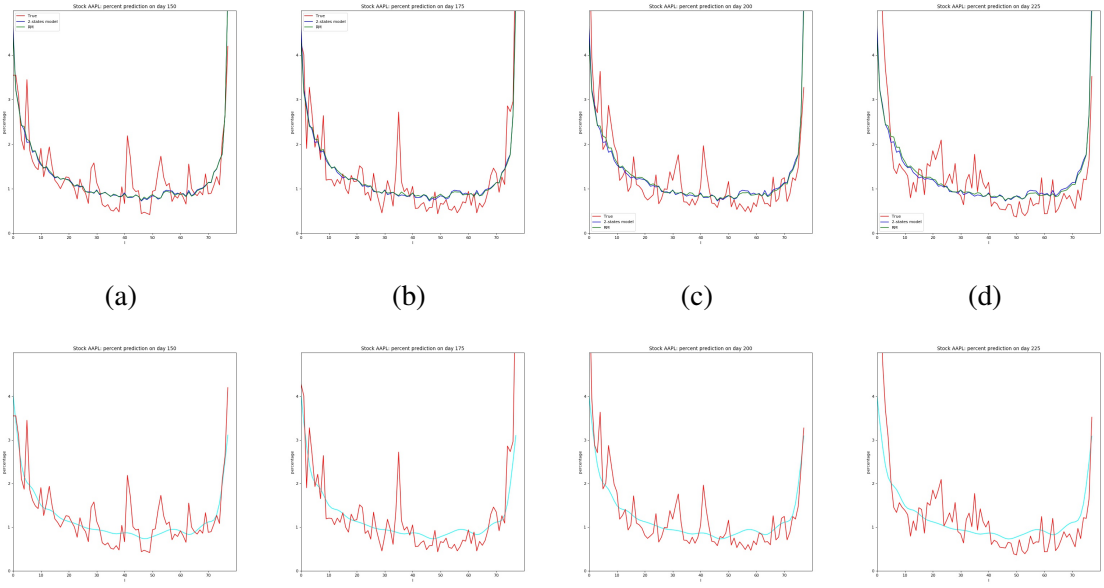
8.1.4 MAPE Distribution

We obtain the distributions of MAPE of 978 stocks from three models and show them in Figure 8.3. We see that our v-state model outperforms baselines by giving smaller MAPEs.



(a) RM (b) Two-state (c) our model: v-state

Figure 8.3: Comparison of MAPE



(a) (b) (c) (d)

(e) (f) (g) (h)

Figure 8.4: comparison of prediction: (a) to (d):baseline models on AAPL, (e) to (h):our v-state model on stock ”AAPL”

8.1.4 Predictions on Specific Days

To better visualize comparisons, we pick ten stocks out of the data set and show their predictions on day 150, 175, 200 and 225. Due to space limitation, we only show one stock ”AAPL” here in Figure 8.4. We see that two-state model almost overlaps RM model. Our v-state model provides a smoother prediction.

8.1.4 *Analysis of v-state Model*

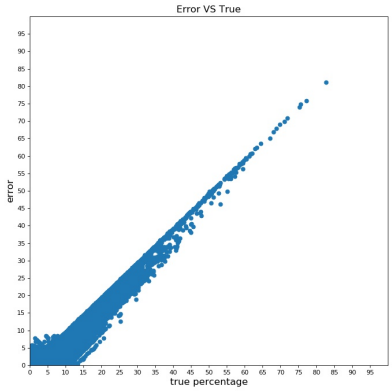


Figure 8.5: Relationship between error and true percentage

In order to further test the precision of our model, we also investigate the relationship between absolute error and true percentage for all stocks. The absolute error of stock s on τ-th bin is defined as

$$\text{error} = |p_{s,\tau}^{\text{predicted}} - p_{s,\tau}^{\text{true}}|. \tag{8.9}$$

As illustrated in Figure 8.5, we present error versus $p_{s,\tau}^{\text{true}}$ for all $978 \times 125 \times 78$ samples. We see there is a nearly linear relationship between absolute error and true percentage. When $p_{s,\tau}^{\text{true}}$ gets larger, the slope gets closer to 1. Moreover, we observe that 95% samples fall into the corner around the original point. For these samples, considering the linearity, mathematically we have

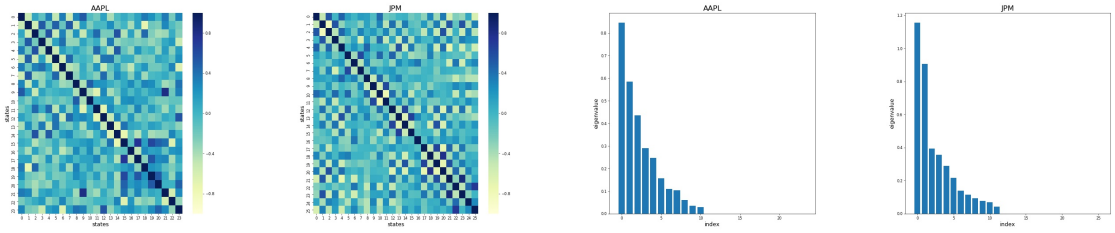
$$\begin{aligned} 0 \leq |p_{s,\tau}^{\text{predicted}} - p_{s,\tau}^{\text{true}}| \leq p_{s,\tau}^{\text{true}}, \\ 0 \leq p_{s,\tau}^{\text{predicted}} \leq 2p_{s,\tau}^{\text{true}}. \end{aligned} \tag{8.10}$$

And for those samples outside of the corner we plug in the linear equation with slope

rate of 1:

$$|p_{s,\tau}^{\text{predicted}} - p_{s,\tau}^{\text{true}}| \approx p_{s,\tau}^{\text{true}},$$
$$p_{s,\tau}^{\text{predicted}} \ll p_{s,\tau}^{\text{true}} \quad \text{or} \quad p_{s,\tau}^{\text{predicted}} \approx 2p_{s,\tau}^{\text{true}}. \tag{8.11}$$

From Equation 8.10 and Equation 8.11, we see that our model provides a lower bound as well as a upper bound for the prediction precision. This is beneficial for the future analysis since it is still not a trivial task to fully capture the movements of trading volume, due to the high-noisy property of original data. It also could be one potential direction in the future.



(a) AAPL (b) JPM (c) AAPL (d) JPM

Figure 8.6: Correlation matrix of hidden states, eigenvalues of transition covariance matrix, "AAPL" and "JPM"

Finally we show complementary examples of correlation matrix of hidden states, eigenvalues of transition covariance matrix in Figure 8.6. The figures suggest that most of states do not highly correlate to each other, meanwhile a few states share linear relationships. We find that half of the eigenvalues are very close to 0. Notice that the eigenvalues from a covariance matrix inform us the directions that the data has different level of spread, the eigenvectors of the covariance matrix indicate the directions that contain more or less data information. The information of correlation matrix and covariance matrix could help us further reduce the number of hidden states which in turn brings less computation. We also leave this as one of future directions.

8.1.5 Conclusion

On the basis of Kalman Filter, we develop a new method to determine the dimension of hidden state for each distinct stock. Our method provides smoothing predictions of intra-day trading volume, brings potential of using gradient based method for further analysis. Through experiments we demonstrate that v-state model gains better prediction precision than two-state model and RM model.

8.2 Mean Field Game Generative Adversarial Network

8.2.1 Introduction

Over the past few years, we have witnessed a great success of generative adversarial networks (GANs) in various applications. Speaking of general settings of GANs, we have two neural networks, one is generator and the other one is discriminator. The goal is to make the generator able to produce samples from noise that matches the distribution of real data. During training process, the discriminator distinguishes the generated sample and real samples while the generator tries to 'fool' the discriminator until an equilibrium state that the discriminator cannot tell any differences between generated samples and real samples.

Wasserstein GAN (WGAN)[7] makes a breakthrough on computation performance since it adopts a more natural distance metric (Wasserstein distance) other than the Total Variation (TV) distance, the Kullback-Leibler (KL) divergence or the Jensen-Shannon (JS) divergence. However most algorithms related to WGAN are based on the Kantorovich duality form of OT (optimal transport) problem, in which we are facing Lipschitz-1 challenge in training process. Though several approximated methods have been proposed to realize the Lipschitz condition in computation, we aim to find a more precise mathematical method.

MFGs[232, 233] are problems that model large populations of interacting agents. They have been widely used in economics, finance and data science. In mean field games, a

continuum population of small rational agents play a non-cooperative differential game on a time horizon [0, T]. At the optimum, the agents reach a Nash equilibrium (NE), where they can no longer unilaterally improve their objectives. In our case the NE leads to a PDE system consisting of continuity equation and Hamilton-Jacobi equation which represent the dynamical evolutions of the population density and the cost value respectively. The PDE system follows a similar pattern within the framework of OT [49], here the curse of infinite dimensionality in MFGs is overcome by applying Hopf formula in density space.

Utilizing Hopf formula to expand MFGs in density space is introduced in [234]. Base on their work, we show that under suitable choice of functionals, the MFGs framework can be reformulated to a Wasserstein-p primal-dual problem. We then handle high dimensional cases by leveraging deep neural networks. Notice that our model is a widely-generalized framework, different choices of Hamiltonian and functional settings lead to different cases. We demonstrate our approach in the context of a series of synthetic and real-world data sets. This chapter is organized as following: 1) In section 8.2.2, we review some typical GANs as well as concepts of MFGs. 2) In section 8.2.3, we develop a new GAN formulation by applying a special Lagrangian, our model naturally and mathematically avoids Lipschitz-1 constraint. 3) In subsection 8.2.4, we empirically demonstrate the effectiveness of our framework through several experiments.

8.2.2 Related Works

8.2.2 Original GAN

Given a generator network G_θ and a discriminator network D_ω, the original GAN objective is to find an mapping that maps a random noise to points that follow an expected distribution, it is written as a minimax problem

$$\inf_{G_\theta} \sup_{D_\omega} \mathbb{E}_{x\sim\rho_{\text{data}}}\big[\log D_\omega(x)\big] + \mathbb{E}_{z\sim\rho_{\text{noise}}}\big[\log\left(1 - D_\omega(G(z))\right)\big]. \tag{8.12}$$

The loss function is derived from the cross-entropy between the real and generated distributions. Since we never know the ground truth distribution, when compared with other generative modeling approaches, GAN has lots of computational advantages. In GAN framework, it is easy to do sampling without requiring Markov chain, we utilize expectations which is also easy to be computed during the process of constructing differentiable loss function. Finally, the framework naturally combines the advantages of neural networks, avoiding curse of dimensionality and is able to be extended to various GANs with different objective functions. The disadvantage of original GAN is that the training process is delicate and unstable due to the reasons which are theoretically investigated in [235].

8.2.2 Wasserstein GAN

Wasserstein distance provides a better and more natural measure to evaluate the distance between two distributions[15]. The objective function of Wasserstein GAN (WGAN) is constructed on the basis of Kantorovich-Rubinstein duality form [6] for distributions ρ_1 and ρ_2:

$$W(\rho_1, \rho_2) = \sup_{\phi, \|\nabla\phi\| \leq 1} \mathbb{E}_{x\sim\rho_1}[\phi(x)] - \mathbb{E}_{x\sim\rho_2}[\phi(x)]. \tag{8.13}$$

Given duality form, one would like to rewrite the duality form in a GAN framework by setting generator G_θ, discriminator D_ω and then minimizing the Wasserstein distance between target distribution and generated distribution:

$$\inf_{G_\theta} \sup_{D_\omega, \|\nabla D_\omega \leq 1\|} \mathbb{E}_{x\sim\rho_{data}}[D_\omega(x)] - \mathbb{E}_{z\sim\rho_{\text{noise}}}[D_\omega(G_\theta(z))]. \tag{8.14}$$

Though WGAN has made a great success in various applications, it brings a well-known Lipschitz constraint problem, namely, we need to make our discriminator satisfies the condition $\|\nabla D_\omega \leq 1\|$. Lipschitz constraint in WGAN has drawn a lot of attentions of researchers, several methods such as weight clipping[7], adding gradient penalty as regu-

larizer[236], applying weight normalization[237] and spectral normalization[238]. These methods facilitate training process, however, all of these methods are approximation ones, besides, we seek for a new formulation to avoid Lipschitz constraint in WGAN theoretically.

8.2.2 Other GANs

A variety of other GANs have been proposed in recent years. These GANs enlarge the application areas and provide great insights of designing generators and discriminators to facilitate the training process. One line of the work is focused on changing objective function, such as f-GAN[239], Conditional-GAN[240], LS-GAN[241] and Cycle-GAN[242]. Another line is focused on developing stable structure of generator and discriminator, including DCGAN[243], Style-GAN[244] and Big-GAN[245]. Though these GANs have been widely applied and accomplished a great success, there is no ”best” GAN[246].

8.2.2 Hamilton-Jacobi Equation and Mean Field Games

Hamilton-Jacobi equation in density space (HJD) plays an important role in optimal control problems[51, 247, 248]. HJD determines the global information of the system[249, 250] and describes the time evolution of the optimal value in density space. In applications, HJD has shown effectiveness in modeling population differential games, also known as MFGs which study strategic dynamical interactions in large populations by setting up finite players' differential games[233, 232]. Within this context, the agents reach a Nash equilibrium at the optimum, where they can no longer unilaterally improve their objectives. The solutions to these problems are obtained by solving the following partial differential

equations (PDEs)[234]:

$$\begin{cases} \partial_t \rho(x,t) + \nabla_x \cdot (\rho(x,t) \nabla_p H(x, \nabla_x \Phi(x,t))) = 0 \\ \partial_t \Phi(x,t) + H(x, \nabla_x \Phi(x,t)) + f(x, \rho(.,t)) = 0 \\ \rho(x,T) = \rho(x), \quad \Phi(x,0) = g(x, \rho(.,0)). \end{cases} \tag{8.15}$$

where $\rho(x,t)$ represents the population density of individual x at time t. $\Phi(x,t)$ represents the velocity of population. We also have each player's potential energy f and terminal cost g. The Hamiltonian H is defined as

$$H(x,p) = \sup_v \{\langle v, p \rangle - L(x,v)\}. \tag{8.16}$$

where L can be freely chosen.

Furthermore, a game is called a potential game when there exists a differentiable potential energy F and terminal cost G, such that the MFG can be modeled as an optimal control problem in density space[251, 234]

$$U(T,\rho) = \inf_{\rho,v} \{ \int_0^T [\int_X L(x, v(x,t)) \rho(x,t) dx - F(\rho(.,t))] dt + G(\rho(.,0))\}. \tag{8.17}$$

where the infimum is taken among all vector fields $v(x,t)$ and densities $\rho(x,t)$ subject to the continuity equation:

$$\begin{cases} \partial_t \rho(x,t) + \nabla \cdot (\rho(x,t) v(x,t)) = 0 \quad 0 \le t \le T \\ \rho(x,T) = \rho(x). \end{cases} \tag{8.18}$$

Our method is derived from this formulation, the details will be introduced in next section.

8.2.2 *Optimal Transport*

To better understand the connections between MFGs and WGAN, we also would like to recap optimal transport(OT). OT rises as a popular topic in recent years, OT-based theories and formulations have been widely used in fluid mechanics[252], control[11], GANs[7] as well as PDE[47]. Recall Monge's problem[6]:

$$\inf_{F} \int_{X} c(x, F(x)) d\mu(x). \tag{8.19}$$

However, the solution of Monge problem may not exist and even if the solution exists, it may not be unique. Recall Kantorovich's problem[24] reads

$$\begin{cases} \inf_{\pi \in \mathcal{P}(X \times Y)} \int_{X \times Y} c(x, y) d\pi(x, y) \\ \int_{Y} d\pi(x, y) = d\mu(x), \quad \int_{X} d\pi(x, y) = d\nu(y). \end{cases} \tag{8.20}$$

Notably, recap the fluid dynamic formulation of OT[24] reads

$$\begin{cases} \inf_{v,\rho} \int_0^T \int_X \frac{1}{2} \|v(x,t)\|^2 \rho(x,t) dx dt \\ \frac{\partial \rho}{\partial t} + \nabla \cdot (\mu \nu) = 0 \\ \rho(x, 0) = \mu, \quad \rho(x, T) = \nu. \end{cases} \tag{8.21}$$

where $v(x,t)$ is the velocity vector. We solve this problem by applying Lagrange multiplier $\Phi(x,t)$ and optimality conditions, finally we have its dual form

$$\begin{cases} \sup_{\Phi,\rho} \int \Phi(x,T)\rho(x,T) - \int \Phi(x,0)\rho(x,0) \\ \frac{\partial \Phi}{\partial t} + \frac{1}{2} \|\nabla \Phi\|^2 = 0. \end{cases} \tag{8.22}$$

We also refer readers to another related dynamical formulation of OT [253].

8.2.3 Formulation via Perspective of MFG

Let's rewrite (8.17) and (8.18) with defining flux function $m(x,t) = \rho(x,t)v(x,t)$, then we have

$$U(T,\rho) = \inf_{\rho,v}\{\int_0^T [\int_X L(x, \frac{m(x,t)}{\rho(x,t)})\rho(x,t)dx - F(\rho(.,t))]dt + G(\rho(.,0))\}, \tag{8.23}$$

with

$$\begin{cases} \partial_t \rho(x,t) + \nabla \cdot m(x,t) = 0 \quad 0 \le t \le T \\ \rho(x,T) = \rho(x). \end{cases} \tag{8.24}$$

We solve above primal-dual problem by applying Lagrange multiplier $\Phi(x,s)$:

$$\begin{aligned} U(T,\rho) = \inf_{\rho(x,t), m(x,t)} \sup_{\Phi(x,t)} \Bigg\{ \int_0^T \left[\int_X L\left(x, \frac{m(x,t)}{\rho(x,t)}\right)\rho(x,t)dx - F(\rho(.,t)) \right] dt + G(\rho(.,0)) \\ + \int_0^T \int_X (\partial_t \rho(x,t) + \nabla \cdot m(x,t))\Phi(x,t)dxdt \Bigg\}. \end{aligned} \tag{8.25}$$

After integration by parts and reordering it leads to

$$\begin{aligned} U(T,\rho) = \sup_{\Phi(x,t)} \inf_{\rho(x,t)} \Bigg\{ - \int_0^T \int_X \rho(x,t) H(x, \nabla\Phi(x,t))dxdt - \int_0^T \int_X \rho(x,t)\partial_t \Phi(x,t)dxdt \\ - \int_0^T F(\rho(.,t))dt + G(\rho(.,0)) - \int_X \rho(x,0)\Phi(x,0)dx \\ + \int_X \rho(x,t)\Phi(x,t)dx \Bigg\}, \end{aligned} \tag{8.26}$$

where

$$H(x, \nabla\Phi(x,s)) = \sup_v \nabla\Phi \cdot v - L(x,v). \tag{8.27}$$

Now let $L(x,v) = \frac{\|v\|^p}{p}$ where $\|\cdot\|$ can be any norm in Euclidean space. For example, choose $\|\cdot\| = \|\cdot\|_2$, namely, $\|v\| = \|v\|_2 = \sqrt{v_1^2 + v_2^2 + ... + v_n^2}$. For arbitrary norm s, we have the minimizer of 8.27 and corresponding H as

$$\nabla\Phi = \|v\|_s^{p-s} \cdot v^{s-1}, \tag{8.28}$$

$$H(v) = \|v\|_s^p - \frac{\|v\|_s^p}{p} = \frac{1}{q}\|v\|_s^p. \tag{8.29}$$

Furthermore we have

$$\begin{aligned}
\|\nabla\Phi\|_{s'}^q &= (\|v\|_s^{p-s})^q \|v^{s-1}\|_{s'}^q \\
&= [(\int |v|^s dx)^{\frac{p-s}{s}}]^{\frac{p}{p-1}} (\int |v^{s-1}|^{\frac{s}{s-1}} dx)^{\frac{p}{p-1}\frac{s-1}{s}} \\
&= (\int |v|^s dx)^{\frac{1}{s}(p-s)\frac{p}{p-1} + \frac{1}{s}(s-1)\frac{p}{p-1}} \\
&= (\int |v|^s dx)^{\frac{p}{s}} \\
&= \|v\|_s^p, \qquad (8.30) \\
H(\nabla\Phi) &= \frac{1}{q}\|\nabla\Phi\|_{s'}^q \quad \text{where} \quad \frac{1}{p} + \frac{1}{q} = 1, \quad \frac{1}{s} + \frac{1}{s'} = 1. \qquad (8.31)
\end{aligned}$$

Now we are able to plug $H(\nabla\Phi)$ back into 8.26, let $T = 1$ and make F and G as 0, then

$$\begin{aligned}
U(1,\rho) = \sup_{\Phi(x,t)} \inf_{\rho(x,t)} \Bigg\{ &\int_0^1 \int_X -(\partial_s\Phi(x,t) + \frac{1}{q}\|\nabla\Phi\|_{s'}^q)\rho(x,t)dxdt \\
&- \int_X \rho(x,0)\Phi(x,0)dx + \int_X \rho(x,1)\Phi(x,1)dx \Bigg\}. \qquad (8.32)
\end{aligned}$$

Next, we extend (8.32) to a GAN framework. We let $\rho(z,t)$ be the generator that pushforward Gaussian noise distribution to target distribution. Specially, we set ρ_1 as the distribution of our ground truth data. We treat $\Phi(x,t)$ as discriminator thus it leads to our

objective function:

$$L_{\text{MFG-GAN}} = \inf_{\rho(x,t)} \sup_{\Phi(x,t)} \Bigg\{ - \mathbb{E}_{z\sim p(z), t\sim Unif[0,1]}[\partial_t \Phi(\rho(z,t),t) + \frac{1}{q}\|\nabla \Phi(\rho(z,t),t)\|_{s'}^q] + \mathbb{E}_{x\sim\rho_1}[\Phi(x,1)] - \mathbb{E}_{z\sim p(z)}[\Phi(\rho(z,0),0)] \Bigg\}. \tag{8.33}$$

We solve this minimax problem by Algorithm 14. Notably, if we extend (8.33) to a general case with a general Hamiltonian H, we revise 8.33 as

$$L = \inf_{\rho(x,t)} \sup_{\Phi(x,t)} \Bigg\{ - \mathbb{E}_{z\sim p(z), t\sim Unif[0,1]}[\partial_t \Phi(G(z,t),t) + H(x,\Phi,\nabla\Phi)] + \mathbb{E}_{x\sim\rho_1}[\Phi(x,1)] - \mathbb{E}_{z\sim p(z)}[\Phi(\rho(z,0),0)] \Bigg\}. \tag{8.34}$$

Algorithm 14

Require: Initialize generator ρ_θ, discriminator Φ_ω

for ρ_θ steps **do**
  Sample x from real distribution
  Sample z from standard normal distribution
  for Φ_ω steps **do**
    Do gradient ascent on $L_{\text{MFG-GAN}}$ to update Φ_ω
  end for
  Do gradient descent on $L_{\text{MFG-GAN}}$ to update ρ_θ
end for

8.2.3 Formulation via Perspective of OT

Now we are going to derive the same objective function from the perspective of OT. Base on the formulation (8.22), we directly apply Lagrange multiplier $\rho(x,s)$ on the constraint

to reformulate original problem as a saddle scheme:

$$\inf_{\rho}\sup_{\Phi}\left\{\int_0^1\int_X -\left(\frac{\partial\Phi}{\partial t}+\frac{1}{2}\|\nabla\Phi\|^2\right)\rho(x,t)dxdt+\int_X\Phi(x,1)\rho(x,1)dx \right. \\ \left. -\int_X\Phi(x,0)\rho(x,0)dx\right\}. \tag{8.35}$$

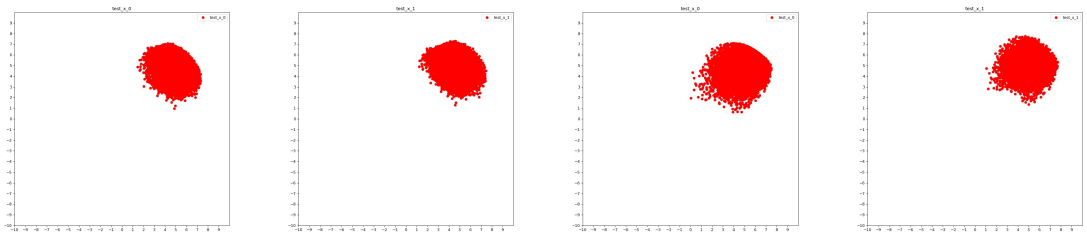
We extend (8.35) to a GAN framework by letting $\rho(z,s)$ be the generator and $\Phi(x,s)$ be the discriminator, thus finally we achieve the same problem:

$$L_{\text{OT-GAN}}=\inf_{\rho(x,t)}\sup_{\Phi(x,t)}\left\{-\mathbb{E}_{z\sim p(z),t\sim Unif[0,1]}[\partial_t\Phi(\rho(z,t),t)+\frac{1}{2}\|\nabla\Phi(\rho(z,t),t)\|^2] \right. \\ \left. +\mathbb{E}_{x\sim\rho_1}[\Phi(x,1)]-\mathbb{E}_{z\sim p(z)}[\Phi(\rho(z,0),0)]\right\}, \tag{8.36}$$

which has the same structure with (8.33), thus (8.33) can also be seen as minimizing Wasserstein-2 distance. A similar formulation is introduced in our recent work [254], where we consider samples from two known distributions and compute their Wasserstein geodesic.

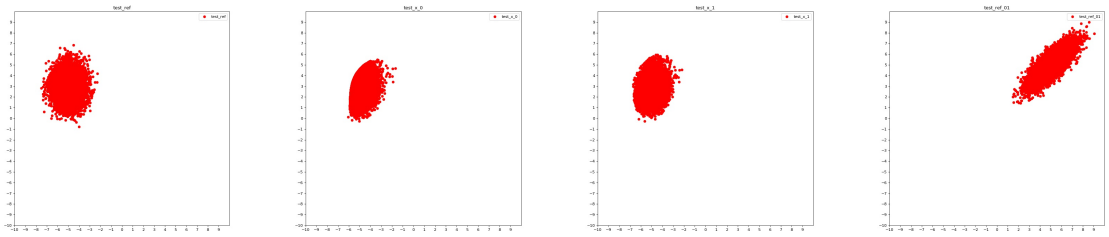
8.2.4 Experiments

Notice that for all experiments we set $s=s'=2$, we adopt fully connected neural networks for both generator and discriminator. In terms of training process, for all synthetic and realistic cases we use the Adam optimizer [55] with learning rate 10^{-4}.

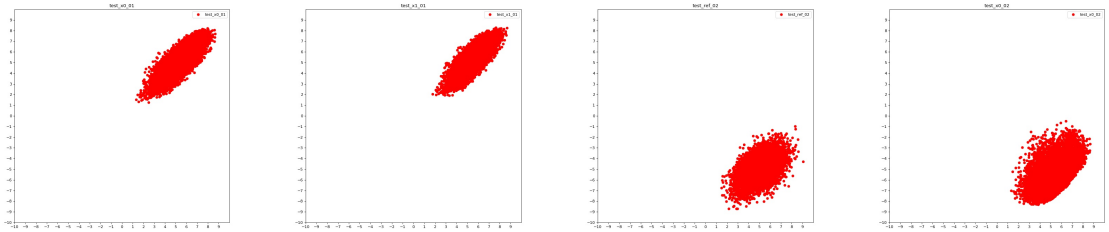


(a) generated by (z,0) (b) generated by (z,1) (c) generated by (z,0) (d) generated by (z,1)

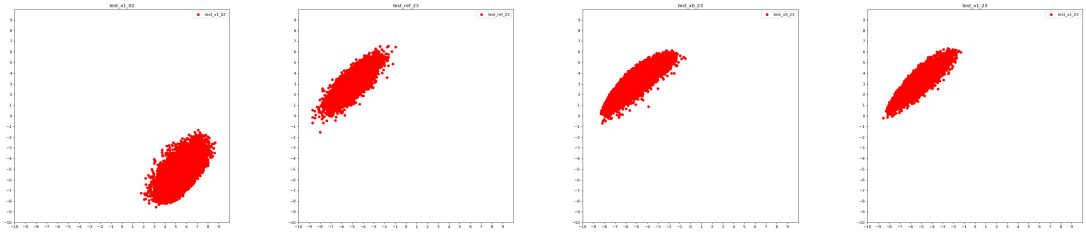
Figure 8.7: Results on 2D synthetic data set



(a) Ground truth (b) generated by (z,0) (c) generated by (z,1) (d) Ground truth



(e) generated by (z,0) (f) generated by (z,1) (g) Ground truth (h) generated by (z,0)



(i) generated by (z,1) (j) Ground truth (k) generated by (z,0) (l) generated by (z,1)

Figure 8.8: Results on 10D synthetic data set

8.2.4 Synthetic

Syn-1: In this case we test our algorithm on 2D case, the target distribution is a Gaussian $\mathcal{N}(\mu, \Sigma)$, where μ=(5,5) and Σ= [[1,0],[0,1]], here we choose $q = 10, p = 1.1$. The generated distributions are shown in Figure 8.7 (a) and (b). **Syn-2**: Here and in all following tests we set $q = 2, p = 2$, in this case we still generate a 2D Gaussian distribution, with the same mean and covariance matrix we used in Syn-1. The generated distributions are shown in Figure 8.7 (c) and (d). **Syn-3**: In this case we generate a 10D Gaussian as our target distribution. Some of 2D projections of the true distributions and generated distributions are shown in Figure 8.8.

(a) iteration 18000 (b) iteration 52000 (c) iteration 146400

Figure 8.9: Generated handwritten digits

8.2.4 Realistic

MNIST: We test our algorithm on MNIST data set, where each sample has 16×16 dimensions. The generated handwritten digits are shown in Figure 8.9. We believe more carefully designed neural networks such as CNN with extra time input dimension will improve the quality of generated pictures.

8.2.5 Conclusion

In this section we derive a new GAN framework based on MFG formulation, by choosing special Hamiltonian and Lagrangian. The same formulation via OT perspective is also presented. The framework avoids Lipschitz-1 constraint and is validated through several synthetic and realistic data sets. We didn't thoroughly investigate the optimal structure of neural networks, we only tried ones with fully connected layers. Some of future works include improving structures of neural networks, or providing a way to embed time input dimension into CNN.

REFERENCES

[1] G. Monge, "Mémoire sur la théorie des déblais et des remblais," *Histoire de l'Académie Royale des Sciences de Paris*, 1781.

[2] L. Kantorovich, "On translation of mass (in russian), c r," in *Doklady. Acad. Sci. USSR*, vol. 37, 1942, pp. 199–201.

[3] J. D. Lafferty, "The Density Manifold and Configuration Space Quantization," *Transactions of the American Mathematical Society*, vol. 305, no. 2, pp. 699–741, 1988.

[4] F. Otto, "The Geometry of Dissipative Evolution Equations: The Porous Medium Equation," *Communications in Partial Differential Equations*, vol. 26, no. 1-2, pp. 101–174, 2001.

[5] J. D. Benamou and Y. Brenier, "A computational fluid mechanics solution to the Monge-Kantorovich mass transfer problem," *Numerische Mathematik*, vol. 84, no. 3, pp. 375–393, 2000.

[6] C. Villani, *Optimal transport: old and new*. Springer Science & Business Media, 2008, vol. 338.

[7] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein gan," in *arXiv: 1701.07875*, 2017.

[8] V. Seguy, B. Damodaran, R. Flamary, R. Courty N., A., and M. Blondel, "Large-scale optimal transport and mapping estimation," *arXiv:1711.02283*, 2017.

[9] V. Krishnan and S. Martínez, "Distributed optimal transport for the deployment of swarms," *IEEE Conference on Decision and Control (CDC)*, pp. 4583–4588, 2018.

[10] D. Inoue, Y. Ito, and H. Yoshida, "Optimal transport-based coverage control for swarm robot systems: Generalization of the voronoi tessellation-based method," *IEEE Control Systems Letters*, vol. 5, no. 4, pp. 1483–1488, 2020.

[11] Y. Chen, T. Georgiou, and M. Pavon, "On the relation between optimal transport and schrödinger bridges: A stochastic control viewpoint," in *Journal of Optimization Theory and Applications*, 2016.

[12] Y. Chen, T. T. Georgiou, and M. Pavon, "Optimal transport in systems and control," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 4, pp. 89–113, 2021.

[13] A. Galichon, *Optimal transport methods in economics*. Princeton University Press, 2018.

[14] E. Hairer, M. Hochbruck, A. Iserles, and C. Lubich, "Geometric numerical integration," *Oberwolfach Reports*, vol. 3, no. 1, pp. 805–882, 2006.

[15] C. Villani, *Topics in optimal transportation*, 58. American Mathematical Soc., 2003.

[16] M. Cuturi, "Sinkhorn distances: Lightspeed computation of optimal transport," in *Advances in Neural Information Processing Systems*, 2013, pp. 2292–2300.

[17] R. Flamary, N. Courty, A. Rakotomamonjy, and D. Tuia, "Optimal transport with laplacian regularization," in *Advances in Neural Information Processing Systems, Workshop on Optimal Transport and Machine Learning*, 2014.

[18] N. Courty, R. Flamary, D. Tuia, and A. Rakotomamonjy, "Optimal transport for domain adaptation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 9, pp. 1853–1865, 2016.

[19] B. Muzellec, R. Nock, G. Patrini, and F. Nielsen, "Tsallis regularized optimal transport and ecological inference," *In Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 31, no. 1, 2017.

[20] A. Dessein, N. Papadakis, and J. Rouas, "Regularized optimal transport and the rot mover's distance," *The Journal of Machine Learning Research*, vol. 19, no. 1, pp. 590–642, 2018.

[21] J. Benamou and Y. Brenier, "A computational fluid mechanics solution to the monge-kantorovich mass transfer problem," *Numerische Mathematik*, vol. 84, no. 3, pp. 375–393, 2000.

[22] C. Finlay, J.-H. Jacobsen, L. Nurbekyan, and A. M. Oberman, "How to train your neural ode: The world of jacobian and kinetic regularization," in *arXiv:2002.02798*, 2020.

[23] L. Yang and G. Karniadakis, "Potential flow generator with L_2 optimal transport regularity for generative models," in *arXiv:1908.11462*, 2019.

[24] J. Benamou, B. Froese, and A. Oberman, "Two numerical methods for the elliptic monge-ampere equation," *ESAIM: Mathematical Modelling and Numerical Analysis*, vol. 44, no. 4, pp. 737–758, 2010.

[25] W. Li, E. Ryu, S. Osher, W. Yin, and W. Gangbo, "A parallel method for earth mover's distance," *Journal of Scientific Computing*, vol. 75, no. 1, pp. 182–197, 2018.

[26] W. Gangbo, W. Li, S. Osher, and M. Puthawala, "Unnormalized optimal transport," *Journal of Computational Physics*, vol. 399, p. 108 940, 2019.

[27] J. Altschuler, J. Niles-Weed, and P. Rigollet, "Near-linear time approximation algorithms for optimal transport via sinkhorn iteration," *Advances in neural information processing systems*, pp. 1964–1974, 2017.

[28] A. Genevay, G. Peyré, and M. Cuturi, "Learning generative models with sinkhorn divergences," *International Conference on Artificial Intelligence and Statistics*, 1608–1617, 2018.

[29] R. Li, X. Ye, H. Zhou, and H. Zha, "Learning to match via inverse optimal transport," in *J. Mach. Learn. Res.*, 2019, 20, pp.80–1.

[30] Y. Xie, X. Wang, R. Wang, and H. Zha, "A fast proximal point method for computing exact wasserstein distance," in *Uncertainty in Artificial Intelligence*, 2020, pp. 433–453.

[31] A. Genevay, M. Cuturi, G. Peyré, and F. Bach, "Stochastic optimization for large-scale optimal transport," in *Advances in neural information processing systems*, 2016, pp. 3440–3448.

[32] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of wasserstein gans," in *Advances in Neural Information Processing Systems*, 2017, pp. 5767–5777.

[33] I. Tolstikhin, O. Bousquet, S. Gelly, and B. Schoelkopf, "Wasserstein auto-encoders," in *arXiv:1711.01558*, 2017, pp. 3440–3448.

[34] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, "Spectral normalization for generative adversarial networks," *arXiv:1802.05957*, 2018.

[35] A. Tong Lin, W. Li, S. Osher, and G. Montúfar, "Wasserstein proximal of gans," 2018.

[36] Y. Dukler, W. Li, A. Tong Lin, and G. Montúfar, "Wasserstein of wasserstein loss for learning generative models," 2019.

[37] Y. Xie, M. Chen, H. Jiang, T. Zhao, and H. Zha, "On scalable and efficient computation of large scale optimal transport," in *In International Conference on Machine Learning*, 2019, pp. 6882–6892.

[38] B. Amos, L. Xu, and J. Kolter, "Sinkhorn distances: Lightspeed computation of optimal transport," in *International Conference on Machine Learning*, 2017, pp. 146–155.

[39] A. Korotin, V. Egiazarian, A. Asadulaev, A. Safin, and E. Burnaev, "Wasserstein-2 generative networks," *arXiv:1909.13082*, 2019.

[40] J. Fan, A. Taghvaei, and Y. Chen, "Scalable computations of wasserstein barycenter via input convex neural networks," *arXiv:2007.04462*, 2020.

[41] A. Makkuva A.and Taghvaei, S. Oh, and J. Lee, "Optimal transport mapping via input convex neural networks," in *International Conference on Machine Learning*, 2020, pp. 6672–6681.

[42] L. Ruthotto, S. J. Osher, W. Li, L. Nurbekyan, and S. W. Fung, "A machine learning framework for solving high-dimensional mean field game and mean field control problems," *Proceedings of the National Academy of Sciences*, vol. 117, no. 17, pp. 9183–9193, 2020.

[43] C. Finlay, A. Gerolin, A. Oberman, and A. Pooladian, "Learning normalizing flows from entropy-kantorovich potentials," in *arXiv:2006.06033*, 2020.

[44] W. Grathwohl, R. Chen, J. Bettencourt, I. Sutskever, and D. Duvenaud, "Ffjord: Free-form continuous dynamics for scalable reversible generative models," in *arXiv: 1810.01367*, 2018.

[45] G. Trigila and E. Tabak, "Data-driven optimal transport," *Communications on Pure and Applied Mathematics*, vol. 69, no. 4, pp. 613–648, 2016.

[46] M. Kuang and E. Tabak, "Sample-based optimal transport and barycenter problems," *Communications on Pure and Applied Mathematics*, vol. 72, no. 8, pp. 1581–1630, 2019.

[47] S. Ma, S. Liu, H. Zha, and H. Zhou, "Learning stochastic behaviour of aggregate data," in *arXiv:2002.03513*, 2020.

[48] I. Yang, "Wasserstein distributionally robust stochastic control: A data-driven approach," *IEEE Transactions on Automatic Control*, 2020.

[49] A. Lin, S. Fung, W. Li, L. Nurbekyan, and S. Osher, "Apac-net: Alternating the population and agent control via two neural networks to solve high-dimensional stochastic mean field games," in *arXiv:2002.10113*, 2020.

[50] H. W. Kuhn and A. W. Tucker, "Nonlinear programming," in *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability*, Berkeley, Calif.: University of California Press, 1951, pp. 481–492.

[51] J. Benamou and Y. Brenier, "A computational fluid mechanics solution to the monge-kantorovich mass transfer problem," in *Numerische Mathematik*, 2000.

[52] M. Kuang and E. G. Tabak, "Preconditioning of optimal transport," *SIAM Journal on Scientific Computing*, vol. 39, no. 4, A1793–A1810, 2017.

[53] E. Reinhard, M. Ashikhmin, B. Gooch, and P. Shirley, "Color transfer between images," *IEEE Comput. Graph. Appl.*, vol. 21, no. 5, pp. 34–41, Sep. 2001.

[54] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[55] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv:1412.6980*, 2014.

[56] Y. Mroueh, C. Li, T. Sercu, A. Raj, and Y. Cheng, "Sobolev gan," in *arXiv:1711.04894*, 2020.

[57] S. Liu, S. Ma, Y. Chen, H. Zha, and H. Zhou, "Learning high dimensional wasserstein geodesics," in *arXiv:2102.02992*, 2021.

[58] J.-C. Hütter and P. Rigollet, *Minimax estimation of smooth optimal transport maps*, 2020. arXiv: 1905.05828 [math.ST].

[59] S. Afriat, "Theory of maxima and the method of lagrange," *SIAM Journal on Applied Mathematics*, vol. 20, no. 3, pp. 343–357, 1971.

[60] L. Rout, A. Korotin, and E. Burnaev, "Generative modeling with optimal transport maps," in *International Conference on Learning Representations*, 2022.

[61] A. Korotin, D. Selikhanovych, and E. Burnaev, "Neural optimal transport," *arXiv:2201.12220*, 2022.

[62] M. Gazdieva, L. Rout, A. Korotin, A. Filippov, and E. Burnaev, "Unpaired image super-resolution with optimal transport maps," *arXiv:2202.01116*, 2022.

[63] Y. Zeng, Z. Lin, H. Lu, and V. M. Patel, "Cr-fill: Generative image inpainting with auxiliary contextual reconstruction," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 14 164–14 173.

[64] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 3730–3738.

[65] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "Gans trained by a two time-scale update rule converge to a local nash equilibrium," *Advances in neural information processing systems*, vol. 30, 2017.

[66] A. Obukhov, M. Seitzer, P.-W. Wu, S. Zhydenko, J. Kyl, and E. Y.-J. Lin, *High-fidelity performance metrics for generative models in pytorch*, version v0.3.0, Version: 0.3.0, DOI: 10.5281/zenodo.4957738, 2020.

[67] M. Cuturi and D. Avis, "Ground metric learning," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 533–564, 2014.

[68] A. Dupuy and A. Galichon, "Personality traits and the marriage market," *Journal of Political Economy*, vol. 122, no. 6, pp. 1271–1319, 2014.

[69] A. Galichon and B. Salanié, "Cupid's invisible hand: Social surplus and identification in matching models," *Available at SSRN 1804623*, 2015.

[70] R. Li, X. Ye, H. Zhou, and H. Zha, "Learning to match via inverse optimal transport.," *Journal of Machine Learning Research*, vol. 20, no. 80, pp. 1–37, 2019.

[71] A. M. Stuart and M.-T. Wolfram, "Inverse optimal transport," *arXiv:1905.03950*, 2019.

[72] A. Dessein, N. Papadakis, and J.-L. Rouas, "Regularized optimal transport and the rot mover's distance," *The Journal of Machine Learning Research*, vol. 19, no. 1, pp. 590–642, 2018.

[73] B. Schmitzer, "Stabilized sparse scaling algorithms for entropy regularized transport problems," *SIAM Journal on Scientific Computing*, vol. 41, no. 3, A1443–A1481, 2019.

[74] G. Peyré and M. Cuturi, "Computational optimal transport," *Foundations and Trends® in Machine Learning*, vol. 11, no. 5-6, pp. 355–607, 2019.

[75] V. Seguy, B. B. Damodaran, R. Flamary, N. Courty, A. Rolet, and M. Blondel, "Large scale optimal transport and mapping estimation," in *International Conference on Learning Representations*, 2018.

[76] A. Genevay, L. Chizat, F. Bach, M. Cuturi, and G. Peyré, "Sample complexity of sinkhorn divergences," *arXiv:1810.02733*, 2018.

[77] G. Huang, C. Guo, M. J. Kusner, Y. Sun, F. Sha, and K. Q. Weinberger, "Supervised word mover's distance," in *Advances in Neural Information Processing Systems*, 2016, pp. 4862–4870.

[78] F. Wang and L. J. Guibas, "Supervised earth mover's distance learning and its computer vision applications," in *European Conference on Computer Vision*, Springer, 2012, pp. 442–455.

[79] P. Zhao and Z.-H. Zhou, "Label distribution learning by optimal transport," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[80] Y. Yang, Y.-F. Wu, D.-C. Zhan, Z.-B. Liu, and Y. Jiang, "Complex object classification: A multi-modal multi-instance multi-label deep network with optimal transport," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, ACM, 2018, pp. 2594–2603.

[81] L. Xu, H. Sun, and Y. Liu, "Learning with batch-wise optimal transport loss for 3d shape recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 3333–3342.

[82] R. Liu, A. Balsubramani, and J. Zou, "Learning transport cost from subset correspondence," *arXiv:1909.13203*, 2019.

[83] A. Bellet, A. Habrard, and M. Sebban, "A survey on metric learning for feature vectors and structured data," *arXiv:1306.6709*, 2013.

[84] T. M. Dagnew and U. Castellani, "Supervised learning of diffusion distance to improve histogram matching," in *International Workshop on Similarity-Based Pattern Recognition*, Springer, 2015, pp. 28–37.

[85] T. Le and M. Cuturi, "Unsupervised riemannian metric learning for histograms using aitchison transformations," in *International Conference on Machine Learning*, 2015, pp. 2002–2011.

[86] S.-i. Amari, R. Karakida, M. Oizumi, and M. Cuturi, "Information geometry for regularized optimal transport and barycenters of patterns," *Neural computation*, vol. 31, no. 5, pp. 827–848, 2019.

[87] G. Aude, M. Cuturi, G. Peyré, and F. Bach, "Stochastic optimization for large-scale optimal transport," *arXiv:1605.08527*, 2016.

[88] P. Dvurechensky, A. Gasnikov, and A. Kroshnin, "Computational optimal transport: Complexity by accelerated gradient descent is better than by sinkhorn's algorithm," in *International conference on machine learning*, PMLR, 2018, pp. 1367–1376.

[89] P. Dvurechensky, A. Gasnikov, S. Omelchenko, and A. Tiurin, "A stable alternative to sinkhorn's algorithm for regularized optimal transport," in *International Conference on Mathematical Optimization Theory and Operations Research*, Springer, 2020, pp. 406–423.

[90] A. Genevay, "Entropy-regularized optimal transport for machine learning," Ph.D. dissertation, Paris Sciences et Lettres, 2019.

[91] H. Janati, B. Muzellec, G. Peyré, and M. Cuturi, "Entropic optimal transport between unbalanced gaussian measures has a closed form," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[92] F.-P. Paty and M. Cuturi, "Regularized optimal transport is ground cost adversarial," in *International Conference on Machine Learning*, PMLR, 2020, pp. 7532–7542.

[93] A. Beck, "On the convergence of alternating minimization for convex programming with applications to iteratively reweighted least squares and decomposition schemes," *SIAM Journal on Optimization*, vol. 25, no. 1, pp. 185–209, 2015.

[94] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, 2009.

[95] P. Cremonesi, Y. Koren, and R. Turrin, "Performance of recommender algorithms on top-n recommendation tasks," in *Proceedings of the fourth ACM conference on Recommender systems*, 2010, pp. 39–46.

[96] A. Mnih and R. R. Salakhutdinov, "Probabilistic matrix factorization," in *Advances in neural information processing systems*, 2008, pp. 1257–1264.

[97] S. Rendle, "Factorization machines with libfm," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 3, no. 3, pp. 1–22, 2012.

[98] A. Paszke *et al.*, "Automatic differentiation in pytorch," in *NIPS-W*, 2017.

[99] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *International Conference on Artificial Intelligence and Statistics*, 2010.

[100] A. Weber, "The usc-sipi image database version 5," *USC-SIPI Report*, vol. 315(1), 1997.

[101] M. Hong, X. Wang, M. Razaviyayn, and Z.-Q. Luo, "Iteration complexity analysis of block coordinate descent methods," *Mathematical Programming*, vol. 163, no. 1-2, pp. 85–114, 2017.

[102] S. Greydanus, M. Dzamba, and J. Yosinski, "Hamiltonian neural networks," *arXiv: 1906.01563*, 2019.

[103] D. Alshamaa, A. Chkeir, F. Mourad-Chehade, and P. Honeine, "Hidden markov model for indoor trajectory tracking of elderly people," in *IEEE Sensors Applications Symposium (SAS)*, 2019.

[104] S. R. Eddy, "Hidden markov models," *Current Opinion in Structural Biology*, vol. 6, no. 3, pp. 361–365, 1996.

[105] F. Farahi and H. S. Yazdi, "Probabilistic kalman filter for moving object tracking," *Signal Processing: Image Communication*, vol. 82, 2020.

[106] A. C. Harvey, *Forecasting, structural time series models and the Kalman filter*. Cambridge University Press, 1990.

[107] R. E. Kalman, "A new approach to linear filtering and prediction problems," *arXiv: 1805.04099*, 1960.

[108] N. P. Santos, V. Lobo, and A. Bernardino, "Unmanned aerial vehicle tracking using a particle filter based approach," in *IEEE Underwater Technology (UT)*, 2019.

[109] P. M. Djuric *et al.*, "Particle filtering," *IEEE Signal Processing Magazine*, vol. 20, no. 5, pp. 19–38, 2003.

[110] M. Deriche, A. A. Absa, A. Amin, and B. Liu, "A novel approach for salt dome detection and tracking using a hybrid hidden markov model with an active contour model," *Journal of Electrical Systems*, vol. 16(3), pp. 276–294, 2020.

[111] Y. Fang, C. Wang, W. Yao, X. Zhao, H. Zhao, and H. Zha, "On-road vehicle tracking using part-based particle filter," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20(12), pp. 4538–4552, 2019.

[112] A. Hefny, C. Downey, and G. J. Gordon, "Supervised learning for dynamical system learning," in *Neural Information Processing Systems*, 2015.

[113] J. Langford, R. Salakhutdinov, and T. Zhang, "Learning nonlinear dynamic models," in *International Conference on Machine Learning*, 2009, pp. 593–600.

[114] T. Hashimoto, D. Gifford, and T. Jaakkola, "Learning population-level diffusions with generative rnns," in *International Conference on Machine Learning*, 2016, pp. 2417–2426.

[115] Y. Wang, B. Dai, L. Kong, S. M. Erfani, J. Bailey, and H. Zha, "Learning deep hidden nonlinear dynamics from aggregate data," in *Uncertainty in Artificial Intelligence*, 2018.

[116] R. Singh, Q. Zhang, and Y. Chen, "Learning hidden markov models from aggregate observations," *arXiv:2011.11236*, 2020.

[117] E. Weinan, J. Han, and A. Jentzen, "Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic

differential equations," in *Communications in Mathematics and Statistics*, 2017, pp. 349–380.

[118] C. Beck, S. Becker, P. Grohs, N. Jaafari, and A. Jentzen, "Solving stochastic differential equations and kolmogorov equations by means of deep learning," 2018.

[119] W. Li, S. Liu, H. Zha, and H. Zhou, "Parametric fokker-planck equation," in *Geometry science of information*, 2019.

[120] B. Øksendal, "Stochastic differential equations," in *Stochastic differential equations*, Springer, 2003, pp. 65–84.

[121] H. Risken and T. Caugheyz, Eds., *The fokker-planck equation: Methods of solution and application*. Springer, 1991.

[122] E. Nelson, *Quantum fluctuations*. Princeton University Press, 1985.

[123] D. Qi and A. Majda, "Low-dimensional reduced-order models for statistical response and uncertainty quantification: Two-layer baroclinic turbulence," *Journal of the Atmospheric Sciences*, vol. 73(12), pp. 4609–4639, 2016.

[124] H. Risken, "The fokker-planck equation," *Springer Series in Synergetics*, vol. 18, pp. 4609–4639, 1989.

[125] Q. Liu and D. Wang, "Stein variational gradient descent: A general purpose bayesian inference algorithm," in *Neural Information Processing Systems*, 2016, pp. 2378–2386.

[126] M. Pavon, E. G. Tabak, and G. Trigila, "The data-driven schroedinger bridge," *arXiv:1806.01364*, 2018.

[127] D. Rezende and S. Mohamed, "Variational inference with normalizing flows," in *arXiv:1505.05770*, 2015.

[128] O. Zienkiewicz and I. Cheung, *The Finite Element Method in Engineering Science*, ser. McGraw-Hill European Publishing Programme. McGraw-Hill, 1971, ISBN: 9780070941380.

[129] J. Sirignano and K. Spiliopoulos, "Dgm: A deep learning algorithm for solving partial differential equations," *Journal of computational physics*, vol. 375, pp. 1339–1364, 2018.

[130] Y. Zang, G. Bao, X. Ye, and H. Zhou, "Weak adversarial networks for high-dimensional partial differential equations," in *arXiv:1907.08272*, 2019.

[131] K. E. Atkinson, *An introduction to numerical analysis*. John wiley & sons, 2008.

[132] P. E. Kloeden and E. Platen, *Numerical solution of stochastic differential equations*. Springer Science & Business Media, 2013, vol. 23.

[133] M. Cuturi, "Sinkhorn distances: Lightspeed computation of optimal transport," in *Advances in Neural Information Processing Systems*, 2013.

[134] Y. Li, "A data-driven method for the steady state of randomly perturbed dynamics," *arXiv:1805.04099*, 2018.

[135] A. Klein *et al.*, "Droplet barcoding for single-cell transcriptomics applied to embryonic stem cells," *Cell*, vol. 161, no. 5, pp. 1187–1201, 2015.

[136] S. C. Hicks, M. Teng, and R. A. Irizarry, "On the widespread and critical impact of systematic bias and batch effects in single-cell rna-seq data," *bioRxiv*, 2015.

[137] R. Chen, Y. Feng, and D. Palomar, "Forecasting intraday trading volume: A kalman filter approach," in *Available at SSRN 3101695*, 2016.

[138] L. S. Pontryagin, *Mathematical theory of optimal processes*. Routledge, 2018.

[139] M. J. Zahr and P.-O. Persson, "An adjoint method for a high-order discretization of deforming domain conservation laws for optimization of flow problems," *Journal of Computational Physics*, vol. 326, pp. 516–543, 2016.

[140] G. N. Milstein and M. V. Tretyakov, Eds., *Stochastic numerics for mathematical physics*. Springer Science & Business Media, 2013.

[141] E. Haier, C. Lubich, and G. Wanner, *Geometric Numerical integration: structure-preserving algorithms for ordinary differential equations*. Springer, 2006.

[142] B. Leimkuhler and S. Reich, *Simulating hamiltonian dynamics*, 14. Cambridge university press, 2004.

[143] G. E. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

[144] H. Abarbanel, *Analysis of observed chaotic data*. Springer Science & Business Media, 2012.

[145] H. Kantz and T. Schreiber, *Nonlinear time series analysis*. Cambridge university press, 2004, vol. 7.

[146] E. Bradley and H. Kantz, "Nonlinear time-series analysis revisited," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 25, no. 9, p. 097 610, 2015.

[147] E. Baake, M. Baake, H. Bock, and K. Briggs, "Fitting ordinary differential equations to chaotic data," *Physical Review A*, vol. 45, no. 8, p. 5524, 1992.

[148] J. Bongard and H. Lipson, "Automated reverse engineering of nonlinear dynamical systems," *Proceedings of the National Academy of Sciences*, vol. 104, no. 24, pp. 9943–9948, 2007.

[149] M. Schmidt and H. Lipson, "Distilling free-form natural laws from experimental data," *science*, vol. 324, no. 5923, pp. 81–85, 2009.

[150] S. L. Brunton, J. L. Proctor, and J. N. Kutz, "Discovering governing equations from data by sparse identification of nonlinear dynamical systems," *Proceedings of the national academy of sciences*, vol. 113, no. 15, pp. 3932–3937, 2016.

[151] G. Tran and R. Ward, "Exact recovery of chaotic systems from highly corrupted data," *Multiscale Modeling & Simulation*, vol. 15, no. 3, pp. 1108–1129, 2017.

[152] H. Schaeffer, G. Tran, and R. Ward, "Extracting sparse high-dimensional dynamics from limited data," *SIAM Journal on Applied Mathematics*, vol. 78, no. 6, pp. 3279–3295, 2018.

[153] F. Lu, M. Zhong, S. Tang, and M. Maggioni, "Nonparametric inference of interaction laws in systems of agents from trajectory data," *Proceedings of the National Academy of Sciences*, vol. 116, no. 29, pp. 14 424–14 433, 2019.

[154] S. H. Rudy, S. L. Brunton, J. L. Proctor, and J. N. Kutz, "Data-driven discovery of partial differential equations," *Science advances*, vol. 3, no. 4, e1602614, 2017.

[155] S. H. Kang, W. Liao, and Y. Liu, "Ident: Identifying differential equations with numerical time evolution," *Journal of Scientific Computing*, vol. 87, no. 1, pp. 1–27, 2021.

[156] P. A. Reinbold, L. M. Kageorge, M. F. Schatz, and R. O. Grigoriev, "Robust learning from noisy, incomplete, high-dimensional experimental data via physically constrained symbolic regression," *Nature communications*, vol. 12, no. 1, pp. 1–8, 2021.

[157] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Multistep neural networks for data-driven discovery of nonlinear dynamical systems," *arXiv:1801.01236*, 2018.

[158] S. H. Rudy, J. N. Kutz, and S. L. Brunton, "Deep learning of dynamics and signal-noise decomposition with time-stepping constraints," *Journal of Computational Physics*, vol. 396, pp. 483–506, 2019.

[159] T. Qin, K. Wu, and D. Xiu, "Data driven governing equations approximation using deep neural networks," *Journal of Computational Physics*, vol. 395, pp. 620–635, 2019.

[160] Z. Long, Y. Lu, X. Ma, and B. Dong, "Pde-net: Learning pdes from data," in *International Conference on Machine Learning*, PMLR, 2018, pp. 3208–3216.

[161] M. Raissi and G. E. Karniadakis, "Hidden physics models: Machine learning of nonlinear partial differential equations," *Journal of Computational Physics*, vol. 357, pp. 125–141, 2018.

[162] C. A. Bailer-Jones, D. J. MacKay, and P. J. Withers, "A recurrent neural network for modelling dynamical systems," *network: computation in neural systems*, vol. 9, no. 4, p. 531, 1998.

[163] Y. Wang, "A new concept using lstm neural networks for dynamic system identification," in *2017 American control conference (ACC)*, IEEE, 2017, pp. 5324–5329.

[164] J. Pathak, B. Hunt, M. Girvan, Z. Lu, and E. Ott, "Model-free prediction of large spatiotemporally chaotic systems from data: A reservoir computing approach," *Physical review letters*, vol. 120, no. 2, p. 024 102, 2018.

[165] S. Mukhopadhyay and S. Banerjee, "Learning dynamical systems in noise using convolutional neural networks," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 30, no. 10, p. 103 125, 2020.

[166] A. Shalova and I. Oseledets, "Tensorized transformer for dynamical systems modeling," *arXiv:2006.03445*, 2020.

[167] S. Greydanus, M. Dzamba, and J. Yosinski, "Hamiltonian neural networks," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[168] T. Bertalan, F. Dietrich, I. Mezić, and I. G. Kevrekidis, "On learning hamiltonian systems from data," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 29, no. 12, p. 121 107, 2019.

[169] Z. Chen, J. Zhang, M. Arjovsky, and L. Bottou, "Symplectic recurrent neural networks," *arXiv:1909.13334*, 2019.

[170] R. Chen and M. Tao, "Data-driven prediction of general hamiltonian dynamics via learning exactly-symplectic maps," in *International Conference on Machine Learning*, PMLR, 2021, pp. 1717–1727.

[171] P. Jin, A. Zhu, G. E. Karniadakis, and Y. Tang, "Symplectic networks: Intrinsic structure-preserving networks for identifying hamiltonian systems," *arXiv:2001.03750*, 2020.

[172] M. Lutter, C. Ritter, and J. Peters, "Deep lagrangian networks: Using physics as model prior for deep learning," *arXiv:1907.04490*, 2019.

[173] P. Toth, D. J. Rezende, A. Jaegle, S. Racanière, A. Botev, and I. Higgins, "Hamiltonian generative networks," *arXiv:1909.13789*, 2019.

[174] Y. D. Zhong, B. Dey, and A. Chakraborty, "Symplectic ode-net: Learning hamiltonian dynamics with control," *arXiv:1909.12077*, 2019.

[175] K. Wu, T. Qin, and D. Xiu, "Structure-preserving method for reconstructing unknown hamiltonian systems from trajectory data," *SIAM Journal on Scientific Computing*, vol. 42, no. 6, A3704–A3729, 2020.

[176] S. Xiong, Y. Tong, X. He, S. Yang, C. Yang, and B. Zhu, "Nonseparable symplectic neural networks," *arXiv:2010.12636*, 2020.

[177] S. Ma, S. Liu, H. Zha, and H. Zhou, "Learning stochastic behaviour of aggregate data," *arXiv:2002.03513*, 2020.

[178] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016, pp. 770–778.

[179] U. Eren and B. Açıkmeşe, "Velocity field generation for density control of swarms using heat equation and smoothing kernels," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 9405–9411, 2017.

[180] S. Zhao, S. Ramakrishnan, and M. Kumar, "Density-based control of multiple robots," in *Proceedings of the 2011 American control conference*, IEEE, 2011, pp. 481–486.

[181] L. C. Pimenta, N. Michael, R. C. Mesquita, G. A. Pereira, and V. Kumar, "Control of swarms based on hydrodynamic models," in *2008 IEEE international conference on robotics and automation*, IEEE, 2008, pp. 1948–1953.

[182] H. Choi, U. Vaidya, and Y. Chen, "A convex data-driven approach for nonlinear control synthesis," *Mathematics*, vol. 9, no. 19, p. 2445, 2021.

[183] Z. Yi, Z. Cao, E. Theodorou, and Y. Chen, "Nonlinear covariance control via differential dynamic programming," in *2020 American Control Conference (ACC)*, IEEE, 2020, pp. 3571–3576.

[184] I. Haasler, Y. Chen, and J. Karlsson, "Optimal steering of ensembles with origin-destination constraints," *IEEE Control Systems Letters*, vol. 5, no. 3, pp. 881–886, 2020.

[185] K. Caluya and A. Halder, "Wasserstein proximal algorithms for the schrödinger bridge problem: Density control with nonlinear drift," *IEEE Transactions on Automatic Control*, 2021.

[186] B. Bonnet and H. Frankowska, "Necessary optimality conditions for optimal control problems in wasserstein spaces," *Applied Mathematics & Optimization*, vol. 84, no. 2, pp. 1281–1330, 2021.

[187] I. Yang, "Wasserstein distributionally robust stochastic control: A data-driven approach," *IEEE Transactions on Automatic Control*, vol. 66, no. 8, pp. 3863–3870, 2020.

[188] A. Hakobyan and I. Yang, "Wasserstein distributionally robust motion control for collision avoidance using conditional value-at-risk," *IEEE Transactions on Robotics*, 2021.

[189] K. Hoshino, "Finite-horizon control of nonlinear discrete-time systems with terminal cost of wasserstein distance," in *2020 59th IEEE Conference on Decision and Control (CDC)*, IEEE, 2020, pp. 4268–4274.

[190] L. Dinh, D. Krueger, and Y. Bengio, "Nice: Non-linear independent components estimation," *arXiv:1410.8516*, 2014.

[191] K. Elamvazhuthi, S. Liu, W. Li, and S. Osher, "Dynamical optimal transport of nonlinear control-affine systems,"

[192] H. Gao, W. Lee, W. Li, Z. Han, S. Osher, and H. V. Poor, "Energy-efficient velocity control for massive numbers of rotary-wing uavs: A mean field game approach," in *GLOBECOM 2020-2020 IEEE Global Communications Conference*, IEEE, 2020, pp. 1–6.

[193] C. Frederick, M. Egerstedt, and H. Zhou, "Collective motion planning for a group of robots using intermittent diffusion," *Journal of Scientific Computing*, vol. 90, no. 1, pp. 1–20, 2022.

[194] Z. Feng, G. Hu, Y. Sun, and J. Soon, "An overview of collaborative robotic manipulation in multi-robot systems," *Annual Reviews in Control*, vol. 49, pp. 113–127, 2020.

[195] P. Zhu, W. Dai, W. Yao, J. Ma, Z. Zeng, and H. Lu, "Multi-robot flocking control based on deep reinforcement learning," *IEEE Access*, vol. 8, pp. 150 397–150 406, 2020.

[196] C. E. Luis, M. Vukosavljev, and A. P. Schoellig, "Online trajectory generation with distributed model predictive control for multi-robot motion planning," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 604–611, 2020.

[197] L. Wang, A. D. Ames, and M. Egerstedt, "Safety barrier certificates for collisions-free multirobot systems," *IEEE Transactions on Robotics*, vol. 33, no. 3, pp. 661–674, 2017.

[198] D. Onken, L. Nurbekyan, X. Li, S. W. Fung, S. Osher, and L. Ruthotto, "A neural network approach for high-dimensional optimal control," *arXiv:2104.03270*, 2021.

[199] *The georgia tech systems research lab (gtsr)*, https://fumin.ece.gatech.edu/index.html.

[200] S. Jansen, *Hands-On Machine Learning for Algorithmic Trading: Design and implement investment strategies based on smart algorithms that learn from data using Python*. Packt Publishing Ltd, 2018.

[201] Q. Kang, H. Zhou, and Y. Kang, "An asynchronous advantage actor-critic reinforcement learning method for stock selection and portfolio management," in *Proceedings of the 2nd International Conference on Big Data Research*, 2018, pp. 141–145.

[202] J. De Spiegeleer, D. B. Madan, S. Reyners, and W. Schoutens, "Machine learning for quantitative finance: Fast derivative pricing, hedging and fitting," *Quantitative Finance*, vol. 18, no. 10, pp. 1635–1643, 2018.

[203] D. Lv, S. Yuan, M. Li, and Y. Xiang, "An empirical study of machine learning algorithms for stock daily trading strategy," *Mathematical problems in engineering*, vol. 2019, 2019.

[204] B. Huang, Y. Huan, L. D. Xu, L. Zheng, and Z. Zou, "Automated trading systems statistical and machine learning methods and hardware implementation: A survey," *Enterprise Information Systems*, vol. 13, no. 1, pp. 132–144, 2019.

[205] K. B. Hansen, "The virtue of simplicity: On machine learning models in algorithmic trading," *Big Data & Society*, vol. 7, no. 1, p. 2 053 951 720 926 558, 2020.

[206] L. Cao, "Ai in finance: A review," *Available at SSRN 3647625*, 2020.

[207] L. Gan, H. Wang, and Z. Yang, "Machine learning solutions to challenges in finance: An application to the pricing of financial products," *Technological Forecasting and Social Change*, vol. 153, p. 119 928, 2020.

[208] V. Todorov, I. Dimov, S. Apostolov, and S. Poryazov, "Highly efficient stochastic approaches for computation of multiple integrals for european options," in *Proceedings of Sixth International Congress on Information and Communication Technology*, Springer, 2022, pp. 1–9.

[209] R. Huptas, "Point forecasting of intraday volume using bayesian autoregressive conditional volume models," *Journal of Forecasting*, vol. 38, no. 4, pp. 293–310, 2019.

[210] S. Borovkova and I. Tsiamas, "An ensemble of lstm neural networks for high-frequency stock market classification," *Journal of Forecasting*, vol. 38, no. 6, pp. 600–619, 2019.

[211] P. Ghosh, A. Neufeld, and J. K. Sahoo, "Forecasting directional movements of stock prices for intraday trading using lstm and random forests," *Finance Research Letters*, vol. 46, p. 102 280, 2022.

[212] M. J. Roe, "Stock market short-termism's impact," *University of Pennsylvania Law Review*, pp. 71–121, 2018.

[213] P. Schroeder, R. Dochow, and G. Schmidt, "Optimal solutions for the online time series search and one-way trading problem with interrelated prices and a profit function," *Computers & Industrial Engineering*, vol. 119, pp. 465–471, 2018.

[214] S. Endres and J. Stübinger, "Optimal trading strategies for lévy-driven ornstein–uhlenbeck processes," *Applied Economics*, vol. 51, no. 29, pp. 3153–3169, 2019.

[215] H. Park, M. K. Sim, and D. G. Choi, "An intelligent financial portfolio trading strategy using deep q-learning," *Expert Systems with Applications*, vol. 158, p. 113 573, 2020.

[216] Y. Fang *et al.*, "Universal trading for order execution with oracle policy distillation," *arXiv preprint arXiv:2103.10860*, 2021.

[217] T. Théate and D. Ernst, "An application of deep reinforcement learning to algorithmic trading," *Expert Systems with Applications*, vol. 173, p. 114 632, 2021.

[218] A. Ranaldo and P. Santucci de Magistris, "Trading volume, illiquidity and commonalities in fx markets," 2018.

[219] T.-A. Dinh and Y.-K. Kwon, "An empirical study on importance of modeling parameters and trading volume-based features in daily stock trading using neural networks," in *Informatics*, Multidisciplinary Digital Publishing Institute, vol. 5, 2018, p. 36.

[220] A. Bernales, C. Cañón, and T. Verousis, "Bid–ask spread and liquidity searching behaviour of informed investors in option markets," *Finance Research Letters*, vol. 25, pp. 96–102, 2018.

[221] P.-Y. Hsu, C. Chou, S.-H. Huang, and A.-P. Chen, "A market making quotation strategy based on dual deep learning agents for option pricing and bid-ask spread estimation," in *2018 IEEE international conference on agents (ICA)*, IEEE, 2018, pp. 99–104.

[222] P. Feldhütter and T. K. Poulsen, "What determines bid-ask spreads in over-the-counter markets?" *Available at SSRN 3286557*, 2018.

[223] H. Yang, A. M. Kutan, and D. Ryu, "Volatility information trading in the index options market: An intraday analysis," *International Review of Economics & Finance*, vol. 64, pp. 412–426, 2019.

[224] A. Alkusani, A. Handayani, and Y. F. Rahmadani, "Linkage stock price, trading volume activity, stock returns and trading frequency on bid ask spread," *Innovation Research Journal*, vol. 1, no. 1, pp. 28–33, 2020.

[225] L. Liu and Z. Pan, "Forecasting stock market volatility: The role of technical variables," *Economic Modelling*, vol. 84, pp. 55–65, 2020.

[226] H. Sun and B. Yu, "Forecasting financial returns volatility: A garch-svr model," *Computational Economics*, vol. 55, no. 2, pp. 451–471, 2020.

[227] W. Daadaa, "Bid-ask spread, corporate board and stock liquidity in emergent markets," *African Journal of Economic and Management Studies*, 2021.

[228] C. Brownlees, F. Cipollini, and G. Gallo, "Intra-daily volume modeling and prediction for algorithmic trading," *Journal of Financial Econometrics*, vol. 9, no. 3, pp. 489–518, 2011.

[229] R. Kalman, "A new approach to linear filtering and prediction problems," 1960.

[230] B. Ajinkya and P. Jain, "The behavior of daily stock market trading volume," *Journal of accounting and economics*, vol. 11, no. 4, pp. 331–359, 1989.

[231] R. Shumway and D. Stoffer, "An approach to time series smoothing and forecasting using the em algorithm," *Journal of time series analysis*, vol. 3, no. 4, pp. 253–264, 1982.

[232] O. Gueant, J. Lasry, and P. Lions, "Mean field games and applications," in *In Paris-Princeton lectures on mathematical finance*, 2010.

[233] J. Lasry and P. Lions, "Mean field games," in *Japanese journal of mathematics*, 2007.

[234] Y. Chow, W. Li, S. Osher, and W. Yin, "Algorithm for hamilton–jacobi equations in density space via a generalized hopf formula," *Journal of Scientific Computing*, vol. 80, no. 2, pp. 1195–1239, 2019.

[235] M. Arjovsky and L. Bottou, "Towards principled methods for training generative adversarial networks," *arXiv: 1701.04862*, 2017.

[236] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville, "Improved training of wasserstein gans," in *Advances in Neural Information Processing Systems*, 2017.

[237] T. Salimans and D. Kingma, "Weight normalization: A simple reparameterization to accelerate training of deep neural networks," in *Advances in Neural Information Processing Systems*, 2016.

[238] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, "Spectral normalization for generative adversarial networks," in *arXiv:1802.05957*, 2018.

[239] S. Nowozin, B. Cseke, and R. Tomioka, "F-gan: Training generative neural samplers using variational divergence minimization," in *Advances in Neural Information Processing Systems*, 2016, pp. 271–279.

[240] M. Mirza and S. Osindero, "Conditional generative adversarial nets," in *arXiv:1411.1784*, 2014.

[241] X. Mao, Q. Li, H. Xie, L. R.Y., W. Z., and S. Paul Smolley, "Least squares generative adversarial networks," in *International Conference on Computer Vision*, 2017.

[242] J. Zhu, T. Park, P. Isola, and A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *International Conference on Computer Vision*, 2017.

[243] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," in *arXiv:1511.06434*, 2015.

[244] T. ZKarras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," in *International Conference on Computer Vision and Pattern Recognition*, 2019.

[245] A. Brock, J. Donahue, and K. Simonyan, "A style-based generator architecture for generative adversarial networks," in *arXiv:1809.11096*, 2018.

[246] M. Lucic, K. Kurach, M. Michalski, S. Gelly, and O. Bousquet, "Are gans created equal? a large-scale study," in *Advances in Neural Information Processing Systems*, 2018.

[247] P. Cardaliaguet, F. Delarue, J. Lasry, and P. Lions, "The master equation and the convergence problem in mean field games," in *arXiv:1509.02505*, 2015.

[248] M. Huang, R. Malhamé, and P. Caines, "Large population stochastic dynamic games: Closed-loop mckean-vlasov systems and the nash certainty equivalence principle," in *Communications in Information & Systems*, 2006.

[249] W. Gangbo, T. Nguyen, and A. Tudorascu, "Hamilton-jacobi equations in the wasserstein space," in *Methods and Applications of Analysis*, 2008.

[250] W. Gangbo and A. Swiech, "Existence of a solution to an equation arising from the theory of mean field games," in *Journal of Differential Equations*, 2015.

[251] P. Cardaliaguet, F. Delarue, J.-M. Lasry, and P.-L. Lions, "The master equation and the convergence problem in mean field games," in *The Master Equation and the Convergence Problem in Mean Field Games*, Princeton University Press, 2019.

[252] M. Agueh, B. Khouider, and L. Saumier, "Optimal transport for particle image velocimetry," in *Communications in Mathematical Sciences*, 2016.

[253] W. Li and S. Osher, "Constrained dynamical optimal transport and its Lagrangian formulation," 2018. arXiv: 1807.00937.

[254] S. Liu, S. Ma, Y. Chen, H. Zha, and H. Zhou, "Learning high dimensional wasserstein geodesics," in *arXiv:2102.02992*, 2021.