Wasserstein Adversarial Robustness

by

Kaiwen Wu

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2020

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Deep models, while being extremely flexible and accurate, are surprisingly vulnerable to "small, imperceptible" perturbations known as adversarial attacks. While the majority of existing attacks focus on measuring perturbations under the ℓ_p metric, Wasserstein distance, which takes geometry in pixel space into account, has long been known to be a suitable metric for measuring image quality and has recently risen as a compelling alternative to the ℓ_p metric in adversarial attacks. However, constructing an effective attack under the Wasserstein metric is computationally much more challenging and calls for better optimization algorithms. We address this gap in two ways: (a) we develop an exact yet efficient projection operator to enable a stronger projected gradient attack; (b) we show that the Frank-Wolfe method equipped with a suitable linear minimization oracle works extremely fast under Wasserstein constraints. Our algorithms not only converge faster but also generate much stronger attacks. For instance, we decrease the accuracy of a residual network on CIFAR-10 to 3.4% within a Wasserstein perturbation ball of radius 0.005, in contrast to 65.6% using the previous Wasserstein attack based on an *approximate* projection operator. Furthermore, employing our stronger attacks in adversarial training significantly improves the robustness of adversarially trained models. Our algorithms are applicable to general Wasserstein constrained optimization problems in other domains beyond adversarial robustness.

Acknowledgements

My graduate study at Waterloo is the first time when I start learning to do research in machine learning. As a junior researcher, I am extremely grateful to my supervisor, Yaoliang Yu, for the training that I received. Along the way, Yaoliang provides me with careful guidance and gives me the freedom to choose my own research topics. Yaoliang is always diligent, knowledgeable and willing to offer help. Through out my graduate study, I have learned a lot from him: I dived into multiple areas, developed research skills and cultivated my own research taste. Everything that I learned from him will serve as the basis of my future research.

I also would like to thank Pascal Poupart and Gautam Kamath for reading my thesis and providing valuable comments. Thanks Pascal for always being encouraging.

A special thanks to Gavin and Ruitong for the help during my internship at Borealis AI. Although the internship does not contribute directly to the thesis, it does produce my first conference paper. That really means a lot to me, especially at a time when I accomplished nothing but four rejections after the entire first year. I start to realize I can still achieve something by working hard, which leads to the thesis work.

The machine learning group at Waterloo has created an excellent learning and research environment, which makes the thesis possible. Allen contributed partially to the implementation of this work and pointed out a critical issue in the initial experimental setting. To Jingjing, thanks for the "teapot" gift. I do love The Office. Chatting with you always makes me feel better mentally during such a hard time of pandemic.

Finally, I thank my parents for their unconditional love. I attribute all of my achievements now, if any, to their encouragement and support.

Dedication

To my family.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Deep models are surprisingly vulnerable to adversarial attacks, namely small or even imperceptible perturbations that completely change the prediction (Szegedy et al. 2014). The existence of adversarial examples has raised a lot of security concerns on deep models, and a substantial amount of work has devoted to this emerging field (Goodfellow et al. 2018), including various attacks as well as defences (*e.g.* Carlini et al. 2017; Goodfellow et al. 2015; Kurakin et al. 2017; Madry et al. 2018; Moosavi-Dezfooli et al. 2016; Papernot et al. 2016). Some empirical defences are shown ineffective later under stronger attacks (Athalye et al. 2018a), which has motivated a line of research on certified defences with *provable* guarantees (*e.g.* Cohen et al. 2019; Gowal et al. 2019; Raghunathan et al. 2018; Tjeng et al. 2019; Wong et al. 2018).

The majority of existing work on adversarial robustness focused on the ℓ_p threat model where the perturbation is measured using the ℓ_p norm. However, the ℓ_p norm, despite being computationally convenient, is long known to be a poor proxy for measuring image similarity: two semantically similar images for human perception are not necessarily close under ℓ_p norm, see (*e.g.* Wang et al. 2009) for some astonishing examples. To this end, threat models beyond ℓ_p norms have been proposed, *e.g.* Engstrom et al. (2019) explore geometric transformation to fool deep networks; Laidlaw et al. (2019) use point-wise functions on pixel values to flip predictions; Tramer et al. (2019) study robustness against multiple (ℓ_p) perturbations.

In the same spirit, Wong et al. (2019) recently proposed the Wasserstein threat model, *i.e.*, adversarial examples are subject to a perturbation budget measured by the Wasserstein distance (a.k.a. earth mover's distance, see *e.g.* Peyré et al. 2019). The idea is to *redistribute* pixel mass instead of adjusting each pixel value as in previous ℓ_p threat models.

Examples of Wasserstein adversarial attacks (generated by our algorithm) and comparison to ℓ_2 and ℓ_∞ adversarial attacks are shown in Figure 1.1. A key advantage of the former is that it explicitly captures geometric information in the image space (*i.e.* how mass moves around matters). For example, a slight translation or rotation of an image usually induces small change in the Wasserstein distance, but may change the l_p distance drastically. In addition, Wasserstein distance has played a pivotal role in generative adversarial networks (Arjovsky et al. 2017), computer vision (Rubner et al. 1997), and much beyond (Peyré et al. 2019).

We note that in the last column in Figure 1.1, all adversarial examples seem to change the semantic meaning of the image. Indeed, large perturbation budgets may change the semantic of the inputs, no matter what threat models. There are definitions of adversarial examples that try to mitigate the issue by introducing an "oracle classifier" (*e.g.*, the Bayes classifier) (Suggala et al. 2019). Under the modified definition, perturbations that change the prediction of the "oracle classifier" are not considered as adversarial examples. But this is beyond the scope of this thesis. We focus on solving the computational challenges in constructing Wasserstein adversarial examples. The techniques that we develop may be of independent interest in different settings involving Wasserstein constraints.

1.1 Wasserstein Distance

Wasserstein distance (a.k.a. earth mover's distance) is a metric defined on the space of finite measures with equal total mass (Peyré et al. 2019). To define Wasserstein distance for images, we view them as discrete measures supported on pixel locations. Let $\mathbf{x}, \mathbf{z} \in [0,1]^n$ be two vectorized images such that $\mathbf{1}^\top \mathbf{x} = \mathbf{1}^\top \mathbf{z}$, *i.e.* they have the same total pixel mass.[1] Their Wasserstein distance is defined as:

$$\mathcal{W}(\mathbf{x}, \mathbf{z}) = \min_{\Pi \geq 0} \ \langle \Pi, C \rangle \ \text{ subject to } \ \Pi \mathbf{1} = \mathbf{x}, \Pi^\top \mathbf{1} = \mathbf{z}, \tag{1.1}$$

where $C \in \mathbb{R}^{n \times n}$ is a cost matrix, with C_{ij} representing the cost of transportation from the i-th to the j-th pixel; and $\Pi \in \mathbb{R}^{n \times n}$ is a transportation or coupling matrix, with Π_{ij} representing the amount of mass transported from the i-th to the j-th pixel. Intuitively, Wasserstein distance measures the minimum cost to move mass from $\mathbf{x}$ to $\mathbf{z}$. Unlike usual

[1] We enforce the equal mass restriction such that $\mathbf{z}$ can be transported from $\mathbf{x}$ by *redistributing* the pixel mass, otherwise the Wasserstein distance is undefined. Wasserstein distance with unbalanced mass has been developed recently, but is not the focus of this thesis.

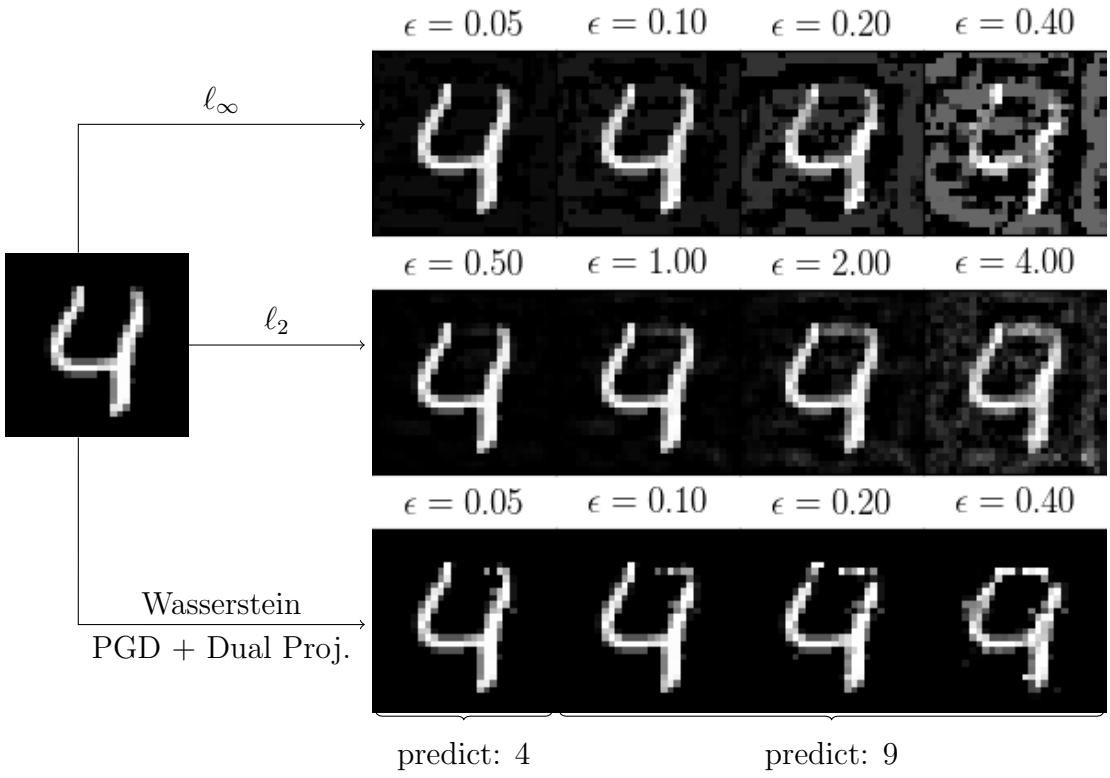


Figure 1.1: ℓ_∞, ℓ_2 and Wasserstein adversarial examples generated by projected gradient descent (PGD) with dual projection. While l_∞ and l_2 norm adversarial examples tend to perturb the background, Wasserstein adversarial examples *redistribute* the pixel mass.

statistical divergences (*e.g.* KL divergence), Wasserstein distance takes the distance between pixels into account hence able to capture the underlying geometry. It has been widely used in statistics, image processing, graphics, machine learning, *etc.*, see the excellent monograph (Peyré et al. 2019).

It not hard to see that (1.1) is a linear program, whose size is *quadratic* w.r.t. the input dimension. For small or medium scale problems, network simplex and interior point methods can handle it well. However, for large scale problems where the linear program involves millions of variables, it is more efficient to solve it approximately. A popular

approximation is through an entropic regularization on the coupling matrix:

$$\begin{aligned} &\text{minimize} \quad \langle \Pi, C \rangle - \lambda H(\Pi) \\ &\text{subject to} \quad \Pi \geq 0,\ \Pi \mathbf{1} = \mathbf{x},\ \Pi^\top \mathbf{1} = \mathbf{z}, \end{aligned} \tag{1.2}$$

where $H(\Pi) = -\sum_{i=1}^{m} \sum_{j=1}^{n} \Pi_{ij} (\log \Pi_{ij} - 1)$ is the entropy of the coupling matrix Π and $\gamma > 0$ is a constant. The entropic regularized formulation (1.2) can be solved iteratively by the celebrated Sinkhorn's fixed point iteration (Cuturi 2013; Sinkhorn 1967). The approximation error introduced by the entropic regularization decays exponentially w.r.t. the inverse of the regularization constant (Cominetti et al. 1994; Weed 2018).

1.2 Wasserstein Adversarial Robustness

Given a deep model that already minimizes some training loss $\mathbf{E}\ell(X, Y; \boldsymbol{\theta})$, we fix (hence also suppress in notation) the model parameter $\boldsymbol{\theta}$ and aim to generate adversarial examples by *maximizing* the loss ℓ subject to some perturbation budget on an input image $\mathbf{x}$. Following Wong et al. (2019), we use the Wasserstein distance to measure perturbations:

$$\begin{aligned} &\underset{\mathbf{z} \in [0,1]^n}{\text{maximize}} \quad \ell(\mathbf{z}, y) \\ &\text{subject to} \quad \mathcal{W}(\mathbf{x}, \mathbf{z}) \leq \delta = \epsilon \mathbf{1}^\top \mathbf{x}, \end{aligned} \tag{1.3}$$

where the perturbation budget δ is proportional to the total mass in the input image $\mathbf{x}$ and ϵ indicates the "proportion". We focus on untargeted attack throughout, where y is the true label of the input image $\mathbf{x}$. All techniques in this thesis can be easily adapted for targeted attacks as well.

In order to optimize (1.3), Wong et al. (2019) develop an approximate projection operator called projected Sinkhorn, and then apply projected gradient ascent to solve (1.3). Here, we give a brief description of the approximate projection method proposed by Wong et al. (2019). The (Euclidean) projection of a vector $\mathbf{w}$ to the Wasserstein ball centered at $\mathbf{x}$ of radius of $\epsilon = \delta$ is

$$\begin{aligned} &\underset{\mathbf{z}}{\text{minimize}} \quad \frac{1}{2} \|\mathbf{w} - \mathbf{z}\|^2 \\ &\text{subject to} \quad \mathcal{W}(\mathbf{x}, \mathbf{z}) \leq \delta, \end{aligned}$$

which is equivalent to the following quadratic program:

$$\begin{aligned} &\underset{\mathbf{z}, \Pi \geq 0}{\text{minimize}} \quad \frac{1}{2} \|\mathbf{w} - \mathbf{z}\|^2 \\ &\text{subject to} \quad \Pi \mathbf{1} = \mathbf{x},\ \Pi^\top \mathbf{1} = \mathbf{z}, \langle \Pi, C \rangle \leq \delta. \end{aligned}$$

Table 1.1: A summary of our proposed algorithms and comparison to projected Sinkhorn. **3rd and 4th column:** Computational complexity for a single iteration of each algorithm (without or with local transportation constraint). n is the dimension of inputs, and k is the local transportation region size (see §2.4.1). **5th column:** The exact convergence rate of projected Sinkhorn is not known yet. **Last column:** Whether the method is an exact or approximate algorithm.

method	optimization space	cost/iter	cost/iter (local)	convergence rate	exact?
projected Sinkhorn (Wong et al. 2019)	image space	$O(n^2)$	$O(nk^2)$	?	✗
dual projection (ours)	coupling matrix	$O(n^2 \log n)$	$O(nk^2 \log k)$	linear	✓
dual linear minimization oracle (ours)	coupling matrix	$O(n^2)$	$O(nk^2)$	linear	✗

For large scale problems, generic quadratic program solvers (*e.g.* interior point methods) are prohibitively expensive especially the projection operator is called in each iteration of projected gradient ascent. Thus, Wong et al. (2019) propose solving the following entropic regularized approximation:

$$\begin{aligned} \underset{\mathbf{z},\Pi\geq 0}{\text{minimize}} \quad & \frac{1}{2}\left\|\mathbf{w}-\mathbf{z}\right\|^2 + \gamma \sum_{i=1}^{n}\sum_{j=1}^{n} \Pi_{ij}\log\Pi_{ij} \\ \text{subject to} \quad & \Pi\mathbf{1}=\mathbf{x},\ \Pi^\top\mathbf{1}=\mathbf{z}, \langle\Pi, C\rangle \leq \delta. \end{aligned} \tag{1.4}$$

The parameter $\gamma > 0$ is the entropic regularization constant. Projected Sinkhorn solves (1.4) through block-coordinate maximization on the dual problem of (1.4).

1.3 Contributions

We end this chapter by summarizing the contributions made by this thesis. Generating Wasserstein adversarial examples requires solving a Wasserstein constrained optimization problem. Wong et al. (2019) developed a projected gradient attack using *approximate* projection (projected Sinkhorn), which we find sometimes too crude and generating suboptimal attacks. In this work, we develop two stronger and faster attacks, based on reformulating the optimization problem (§2.1) and applying projected gradient descent (PGD) and Frank-Wolfe (FW), respectively. For the PGD attack, we design a specialized algorithm to compute the projection *exactly*, which significantly improves the attack quality (§2.2). For FW, we develop a *faster* algorithm to solve the linear minimization step with entropic smoothing (§2.3). Both subroutines enjoy fast linear convergence rates. Synthetic experiments on simple convex functions (Table 3.1) show that both algorithms are able to

converge to high precision solutions. Extensive experiments on large scale datasets (§3.2) confirm the improved quality and speed of our attacks. In particular, for the first time we successfully construct Wasserstein adversarial examples on the ImageNet dataset. A quick comparison of projected Sinkhorn and our algorithms is shown in Table 1.1. Finally, we show that employing our stronger and faster attacks in adversarial training can significantly improve the robustness of adversarially trained models. Our implementation is available at https://github.com/watml/fast-wasserstein-adversarial.

This thesis is an extended version of my published work (Wu et al. 2020) at ICML 2020.

Chapter 2

Algorithms for Wasserstein Adversarial Robustness

In this chapter, we present algorithms for solving the Wasserstein constrained problem (1.3). In §2.1, we propose an alternative formulation for it, which is computationally more appealing . In §2.2 and §2.3, we design efficient projection operator and linear minimization oracle for this new formulation, enabling the usage of projected gradient and Frank-Wolfe methods. In §2.4, we discuss acceleration by exploiting the structured sparsity in the transportation matrix and a method to stabilize the algorithms. In the end, we point out some issues of generated Wasserstein adversarial examples in the prior work and provide an method to fix them.

2.1 Formulation

The difficulty of solving (1.3) lies in the constraint. Even checking the feasibility of $\mathbf{z}$ requires solving a large linear program. Thus, our first step is slightly reformulating (1.3) to simplify the constraint. We expand the constraint in (1.3), and jointly maximize the objective over $\mathbf{z}$ and Π:

$$\begin{aligned} &\underset{\mathbf{z},\Pi\geq 0}{\text{maximize}} \quad \ell(\mathbf{z}, y) \\ &\text{subject to } \Pi\mathbf{1} = \mathbf{x},\ \Pi^\top\mathbf{1} = \mathbf{z},\ \langle \Pi, C\rangle \leq \delta. \end{aligned}$$

Note that we have dropped the domain constraint $\mathbf{z} \in [0,1]^n$, which we will revisit in §2.4.3. We plug in the constraint $\Pi^\top \mathbf{1} = \mathbf{z}$ into the objective to eliminate $\mathbf{z}$:

$$\begin{aligned} &\underset{\Pi \geq 0}{\text{maximize}} \quad \ell(\Pi^\top \mathbf{1}, y) \\ &\text{subject to} \;\; \Pi \mathbf{1} = \mathbf{x}, \;\; \langle \Pi, C \rangle \leq \delta, \end{aligned} \tag{2.1}$$

arriving at a constrained optimization problem w.r.t. Π alone with two linear constraints on it. Yet, problem (1.3) and (2.1) are clearly equivalent. Moreover, problem (2.1) has its own interpretation: instead of maximizing the loss in the image space, it maximizes the loss in the transportation space, searching for a feasible transportation plan Π with cost no greater than δ, to transport $\mathbf{x}$ to an adversarial example. Given a solution Π to (2.1), we generate adversarial examples by summing over the columns of Π, *i.e.*, $\mathbf{x}_{adv} = \Pi^\top \mathbf{1}$. We note that $\nabla_\Pi \ell = \mathbf{1}(\nabla_{\mathbf{x}_{adv}} \ell)^\top$, where $\nabla_{\mathbf{x}_{adv}} \ell$ can be computed efficiently using backpropagation.

We propose PGD and FW to optimize (2.1). While both PGD and FW have been previously used to generate adversarial examples (Chen et al. 2020; Madry et al. 2018), they are based on the ℓ_p threat model that measures image perturbations under the ℓ_p distance. Instead, for the Wasserstein problem in (2.1), applying PGD and FW requires specialized algorithms for the projection operator and the linear minimization oracle, which are the main goals of §2.2 and §2.3.

Throughout the paper, w.l.o.g. we assume that all entries in the cost matrix C are nonnegative and $C_{ij} = 0 \Leftrightarrow i = j$. All common cost matrices satisfy this assumption.

2.2 Projected Gradient with Dual Projection

We apply PGD to maximize (2.1) for generating Wasserstein adversarial examples, and arrive at the following update rule on the coupling matrix Π:

$$\Pi^{(t+1)} = \text{Proj}_{\mathcal{C}} \left(G \right),$$

where

$$G = \Pi^{(t)} + \eta_t \nabla_\Pi \ell\big((\Pi^{(t)})^\top \mathbf{1}, y\big)$$

and $\text{Proj}_{\mathcal{C}}(\cdot)$ denotes the (Euclidean) projection operator onto the convex set $\mathcal{C}$, represented by the constraints in (2.1). Namely, we take a gradient step and then project it back to

Algorithm 1: Dykstra's Projection Algorithm

Input: $G \in \mathbb{R}^{n \times n}$, two convex sets $\mathcal{C}_s$ and $\mathcal{C}_h$
Output: The projection of G to $\mathcal{C}_s \cap \mathcal{C}_h$

$\Pi_h^{(0)} = G$
$I_s^{(0)} = I_h^{(0)} = O$
for $t = 0, 1, \ldots,$ maxiter **do**
  $\Pi_s^{(t+1)} = \mathrm{Proj}_{\mathcal{C}_s}\left(\Pi_h^{(t)} - I_s^{(t)}\right)$
  $I_s^{(t+1)} = \Pi_s^{(t+1)} - \Pi_h^{(t)} + I_s^{(t)}$
  $\Pi_h^{(t+1)} = \mathrm{Proj}_{\mathcal{C}_h}\left(\Pi_s^{(t+1)} - I_h^{(t)}\right)$
  $I_h^{(t+1)} = \Pi_h^{(t+1)} - \Pi_s^{(t+1)} + I_h^{(t)}$
return $\Pi_s^{(t+1)}$

the feasible set. The projection operator $\mathrm{Proj}_{\mathcal{C}}(G)$ is given by the following quadratic program:

$$\begin{aligned} \underset{\Pi \geq 0}{\text{minimize}} \quad & \tfrac{1}{2}\|\Pi - G\|_{\mathrm{F}}^2 \\ \text{subject to} \quad & \Pi \mathbf{1} = \mathbf{x}, \ \langle \Pi, C \rangle \leq \delta. \end{aligned} \tag{2.2}$$

While any quadratic programming solver (*e.g.* interior point method) could be used to solve (2.2), they do not scale well in high dimensions. For high resolution images, (2.2) could involve millions of variables. Since this projection is called in each iteration of PGD, it needs to be solved by a highly efficient algorithm. Below, we exploit the structure of this problem to design a fast specialized projection operator.

2.2.1 A First Attempt: Dykstra's Projection

A simple observation is that the constraint in (2.2) is precisely the intersection of the following two convex sets:

$$\mathcal{C}_s = \left\{\Pi \in \mathbb{R}^{n \times n} : \Pi \geq 0, \ \Pi \mathbf{1} = \mathbf{x}\right\}$$

and

$$\mathcal{C}_h = \left\{\Pi \in \mathbb{R}^{n \times n} : \langle \Pi, C \rangle \leq \delta\right\},$$

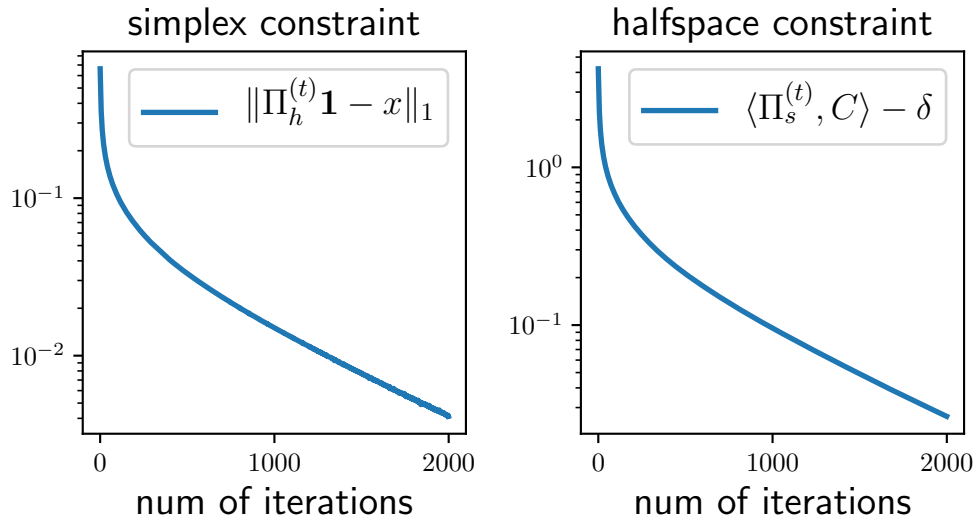


Figure 2.1: Convergence of Dykstra's algorithm. The violation of simplex constraint and halfspace constraint of $\Pi_h^{(t)}$ and $\Pi_s^{(t)}$ (the iterates produced by Dykstra's algorithm) gradually decrease to zero, but at a slow rate.

where each row of $\mathcal{C}_s$ is a simplex constraint, requiring Wassserstein adversarial examples to preserve total mass; and $\mathcal{C}_h$ is a half space constraint, restricting the perturbation budget of Wasserstein adversarial examples. It is known that projection to the intersection of convex sets can be computed by Dykstra's algorithm (Boyle et al. 1986; Dykstra 1983), provided that the projection to each convex set can be computed easily. Hence, our first attempt is to apply Dykstra's algorithm to solve (2.2)

Dykstra's algorithm, applying to these two convex sets, is presented in Algorithm 1. Intuitively, Dykstra's algorithm projects G alternatively to $\mathcal{C}_s$ and $\mathcal{C}_h$ in each iteration. Notice that before projecting to $\mathcal{C}_s$ (or $\mathcal{C}_h$), the increment of the last iteration $I_s^{(t)}$ (or $I_h^{(t)}$) is subtracted from $\Pi_h^{(t)}$ (or $\Pi_s^{(t+1)}$). These increments play a crucial role in the convergence of Dykstra's algorithm. It has been shown that both $\Pi_s^{(t)}$ and $\Pi_h^{(t)}$ converge to the projection of G onto $\mathcal{C}_s \cap \mathcal{C}_h$ (Boyle et al. 1986; Dykstra 1983).

To implement Dykstra's algorithm, we need two subroutines to compute the projection onto $\mathcal{C}_s$ and $\mathcal{C}_h$ respectively. The projection onto $\mathcal{C}_h$ admits a closed form expression:

$$\mathrm{Proj}_{\mathcal{C}_h}(\Pi) = \Pi - \frac{\max\{\langle \Pi, C\rangle - \delta, 0\}}{\|C\|_{\mathrm{F}}^2} C.$$

The projection onto $\mathcal{C}_s$ has an algorithm running in $O(n^2 \log n)$ time, by projecting each

row of G to a simplex. For the simplex projection algorithm, we direct readers to (Duchi et al. 2008).

The convergence rate of Dykstra's algorithm highly relies on the geometry of these two convex sets $\mathcal{C}_s$ and $\mathcal{C}_h$ (Deutsch et al. 1994). In fact, we observe that Dykstra's algorithm converges slowly in some cases. In Figure 2.1, we plot the convergence curves of Dykstra's algorithm for a matrix G of size 100×100. The algorithm seems to converge linearly, but it takes a few hundred iterations before the linear convergence rate kicks in. Besides, the linear rate does not seem to be fast. See Appendix A.3 for more details of this experiment.

2.2.2 Dual Projection

Instead, we develop a dual projection method which converges much faster than Dykstra's algorithm. By the method of Lagrange multipliers, we derive the dual problem:

Proposition 1 *The dual of* (2.2) *is*

$$\underset{\lambda \geq 0}{\text{maximize}} \;\; g(\lambda), \tag{2.3}$$

where

$$g(\lambda) = \min_{\Pi \mathbf{1} = \mathbf{x}, \Pi \geq 0} \; \tfrac{1}{2}\|\Pi - G\|_{\mathrm{F}}^2 + \lambda \left(\langle \Pi, C \rangle - \delta \right). \tag{2.4}$$

In addition, the derivative of $g(\lambda)$ *at a point* $\lambda = \tilde{\lambda}$ *is*

$$g'(\tilde{\lambda}) = \langle \tilde{\Pi}, C \rangle - \delta, \tag{2.5}$$

where

$$\tilde{\Pi} = \operatorname*{argmin}_{\Pi \mathbf{1} = \mathbf{x}, \Pi \geq 0} \; \|\Pi - G + \tilde{\lambda} C\|_{\mathrm{F}}^2. \tag{2.6}$$

Both $g(\lambda)$ *and* $g'(\lambda)$ *can be evaluated in* $O(n^2 \log n)$ *time deterministically for any given* λ.

The derivation of Proposition 1 is based on the observation that (2.4) is equivalent to computing a projection operator $\mathrm{Proj}_{\mathcal{C}_s}(G - \lambda C)$. Since the constraint $\mathcal{C}_s$ is independent for each row, it can be further reduced to projecting each row of $G - \lambda C$ to a simplex. The well-known simplex projection algorithm (*e.g.* Duchi et al. 2008) takes $O(n \log n)$ time, thus the projection to $\mathcal{C}_s$ takes $O(n^2 \log n)$ time.

Although this dual problem does not have a closed form expression, Proposition 1 provides a method to evaluate its objective and gradient, which is sufficient to use first order optimization algorithms. Here, we choose a simple algorithm with linear convergence rate, by exploiting the fact that the dual objective (2.4) is a univariate function. First, we derive an upper bound of the dual solution.

Proposition 2 *The dual solution $\lambda^\star$ of (2.3) satisfies*

$$0 \le \lambda^\star \le \frac{2\left\|\mathrm{vec}(G)\right\|_\infty + \left\|\mathbf{x}\right\|_\infty}{\min_{i\neq j}\{C_{ij}\}}. \tag{2.7}$$

Since $g(\lambda)$ is concave and differentiable, we make the following simple observation: (a) any point $l > 0$ with positive derivative is a lower bound of $\lambda^\star$; (b) any point $u > 0$ with negative derivative is an upper bound of $\lambda^\star$. Thus, we start with the lower bound and upper bound in Proposition 2, and use bisection method to search for $\lambda^\star$, by iteratively testing the sign of the derivative. Eventually, the bisection method converges to either a stationary point or the boundary $\lambda = 0$, which are exactly maximizers in both cases. Since the bisection method halves the gap between lower and upper bounds in each iteration, it converges linearly. Once we solve the dual, we can recover the primal solution by the following.

Proposition 3 *The primal solution $\Pi^\star$ and the dual solution $\lambda^\star$ satisfies*

$$\Pi^\star = \operatorname*{argmin}_{\Pi\mathbf{1}=\mathbf{x},\Pi\ge 0} \left\|\Pi - G + \lambda^\star C\right\|_{\mathrm{F}}^2,$$

thus $\Pi^\star$ can be computed in $O(n^2 \log n)$ time given $\lambda^\star$.

The full dual projection is presented in Algorithm 2, where u is initialized as the upper bound (2.7). A discussion of the stopping criterion in Line 2 is deferred to Appendix A.2.

Finally, when the loss ℓ is convex (concave) in $\mathbf{x}$, it is also convex (concave) in Π after the reformulation in §2.1. In this case, projected gradient with dual projection is *guaranteed* to converge to a global optimum. When ℓ is nonconvex, projected gradient still converges to a stationary point.

Algorithm 2: Dual Projection

Input: $G, C \in \mathbb{R}^{n\times n}$, $\mathbf{x} \in \mathbb{R}^n$, $\delta > 0$, $l = 0$, $u > 0$
Output: $\tilde{\Pi} \in \mathbb{R}^{n\times n}$

while *not converged* **do**
  $\tilde{\lambda} = \frac{1}{2}(l + u)$
  $\tilde{\Pi} = \operatorname{argmin}_{\Pi\mathbf{1}=\mathbf{x}, \Pi\geq 0} \|\Pi - G + \tilde{\lambda}C\|_{\mathrm{F}}^2$
  if $\langle \tilde{\Pi}, C\rangle > \delta$ **then** $l = \tilde{\lambda}$
  else $u = \tilde{\lambda}$

2.3 Frank-Wolfe with Dual LMO

In this section, we apply the Frank-Wolfe algorithm (Frank et al. 1956) to maximize (2.1). During each iteration, FW first solves the following linear minimization problem:

$$\hat{\Pi} = \operatorname*{argmin}_{\Pi\in\mathcal{C}} \langle \Pi, H\rangle, \tag{2.8}$$

where

$$H = -\nabla_{\Pi}\ell\big((\Pi^{(t)})^{\top}\mathbf{1}, y\big)$$

and $\mathcal{C}$ is the convex set represented by constraints in (2.1). Then, we take a convex combination of $\Pi^{(t)}$ and $\hat{\Pi}$:

$$\Pi^{(t+1)} = (1 - \eta_t)\Pi^{(t)} + \eta_t\hat{\Pi}.$$

Step (2.8) is referred as the linear minimization oracle (LMO) in the literature, and is reduced to solving the following linear program in each iteration:

$$\begin{aligned} &\underset{\Pi\geq 0}{\text{minimize}} && \langle \Pi, H\rangle \\ &\text{subject to} && \Pi\mathbf{1} = \mathbf{x},\ \langle \Pi, C\rangle \leq \delta. \end{aligned} \tag{2.9}$$

Standard linear programming solvers do not scale well in high dimensions. Instead, we exploit the problem structure again to design a fast, specialized algorithm for (2.9).

2.3.1 A First Attempt: Optimizing the Dual

Our fist attempt is to extend the idea in §2.2 to the linear minimization step. We first derive an equivalent dual problem via the method of Lagrange multipliers:

Proposition 4 *The dual problem of* (2.9) *is*

$$\underset{\lambda \geq 0}{\text{maximize}} \quad -\lambda\delta + \sum_{i=1}^{n} x_i \min_{1 \leq j \leq n} \Big(H_{ij} + \lambda C_{ij} \Big). \tag{2.10}$$

In addition, we provide an upper bound of the maximizer.

Proposition 5 *The dual solution* $\lambda^\star$ *of* (2.10) *satisfies*

$$0 \leq \lambda^\star \leq \frac{2 \left\| \mathrm{vec}(H) \right\|_\infty}{\min_{i \neq j} \{C_{ij}\}}. \tag{2.11}$$

Unlike dual projection, the dual objective (2.10) here is not differentiable. In fact, it is piecewise linear. Nevertheless, one can still solve it using derivative-free methods such as bisection method on the supergradient, or golden section search on the objective, both of which converge linearly.

However, after obtaining the dual solution, one cannot recover the primal solution easily. Consider the following recovery rule by minimizing the Lagrangian:

$$\Pi^\star \in \underset{\Pi \geq 0, \Pi \mathbf{1} = \mathbf{x}}{\operatorname{argmin}} \ \langle \Pi, H + \lambda^\star C \rangle - \lambda^\star \delta, \tag{2.12}$$

where $\Pi^\star$ and $\lambda^\star$ are primal and dual solutions respectively. There are two issues: (a) there might be infinitely many solutions to (2.12) and it is not easy to determine $\Pi^\star$ among them; (b) even if the solution to (2.12) is unique, a slight numerical perturbation could change the minimizer drastically. In practice, such instability may even result in an infeasible Π, generating *invalid* adversarial examples outside the Wasserstein perturbation budget. We direct readers to Appendix A.4 for a concrete example and further discussions.

2.3.2 Dual LMO via Entropic Regularization

To address the above issues, we instead solve an entropic regularized version of (2.9) as an approximation:

$$\begin{aligned} \underset{\Pi \geq 0}{\text{minimize}} \quad & \langle \Pi, H \rangle + \gamma \sum_{i=1}^{n} \sum_{j=1}^{n} \Pi_{ij} \log \Pi_{ij} \\ \text{subject to} \quad & \Pi \mathbf{1} = \mathbf{x}, \ \langle \Pi, C \rangle \leq \delta, \end{aligned} \tag{2.13}$$

where γ is a regularization parameter. The difference between (2.10) and (2.13) is that the objective is strongly convex with the entropic regularization. For any dual variable, the corresponding primal variable by minimizing the Lagrangian is always unique, which allow us to recover the primal solution from dual easily.

Proposition 6 *The dual problem of* (2.13) *is*

$$\underset{\lambda \geq 0}{\text{maximize}} \; -\lambda\delta + \gamma \sum_{i:x_i>0} x_i \log x_i - \gamma \sum_{i=1}^{n} x_i \log \sum_{j=1}^{n} \exp\left(-\frac{H_{ij}+\lambda C_{ij}}{\gamma}\right). \tag{2.14}$$

The third term in (2.14) is essentially a softmin operator along each row of $H + \lambda C$, by observing that

$$\lim_{\gamma \to 0} -\gamma \log \sum_{j=1}^{n} \exp\left(-\tfrac{H_{ij}+\lambda C_{ij}}{\gamma}\right) = \min_{1 \leq j \leq n} \Big(H_{ij} + \lambda C_{ij}\Big).$$

Thus, from the point view of dual, (2.14) is a smooth approximation of (2.10), recovering (2.10) precisely as $\gamma \to 0$. In our implementation, we use the usual log-sum-exp trick to enhance the numerical stability of the softmin operator.

With entropic regularization, we have the following recovery rule for primal solution and upper bound on dual solution.

Proposition 7 *The primal solution* $\Pi^\star$ *and the dual solution* $\lambda^\star$ *satisfy*

$$\Pi^\star_{ij} = x_i \cdot \frac{\exp\left(-\frac{H_{ij}+\lambda^\star C_{ij}}{\gamma}\right)}{\sum_{j=1}^{n} \exp\left(-\frac{H_{ij}+\lambda^\star C_{ij}}{\gamma}\right)}. \tag{2.15}$$

Proposition 8 *The dual solution* $\lambda^\star$ *of* (2.14) *satisfies*

$$0 \leq \lambda^\star \leq \frac{\left[2\left\|\mathrm{vec}(H)\right\|_\infty + \gamma \log \frac{1}{\delta}\mathbf{x}^\top C \mathbf{1}\right]_+}{\min_{i \neq j}\{C_{ij}\}}. \tag{2.16}$$

Note that the recovery rule (2.15) is essentially applying the softmin activation along each row of $H + \lambda^\star C$. With the upper bound (2.16), we can apply the same technique in §2.2 to solve the dual. In particular, the bisection method on the dual objective results in

an algorithm almost identical to Algorithm 2 with only two exceptions: (a) we replace the upper bound in (2.7) with (2.16) and (b) we replace Line 2 with the primal recover rule (2.15).

Unlike dual projection, the second order derivative of (2.14) can be computed in a closed form, which allows the usage of second order optimization algorithms such as Newton's method for further acceleration. However, we observed that Newton's method fails to converge in some cases. This might because the smooth dual (2.14), as an approximation of (2.10), still behaves similar to a piecewise linear function even after the regularization. Pure Newton's method might easily overshoot in this case. Thus, we choose bisection method due to its simplicity, stability and relatively fast convergence rate. In application domains where the convergence speed of the dual is a concern, it is possible to consider second order methods for potentially faster convergence.

Although both projected Sinkhorn (Wong et al. 2019) and dual LMO use entropic approximation, we emphasize that there are two fundamental differences. First, the entropic regularization in dual LMO does not affect the convergence rate. In contrast, the convergence rate of projected Sinkhorn highly depends on γ. In particular, small γ often slows down the convergence rate empirically. Second, unlike the entropic regularized quadratic program used in projected Sinkhorn, entropic regularized linear program like (2.13) has been well studied. Applications in optimal transport (Cuturi 2013) have demonstrated its empirical success; theoretical guarantees on the exponential decay of approximation error have been established (Cominetti et al. 1994; Weed 2018).

For a thorough discussion on the convergence properties of FW on convex or nonconvex functions and beyond, we direct readers to (Yu et al. 2017).

2.4 Practical Considerations

In this section, we comment on some practical considerations for implementations of algorithms in §2.2 and §2.3.

2.4.1 Acceleration via Exploiting Sparsity

Both algorithms in §2.2 and §2.3 require computation on a full transportation matrix $\Pi \in \mathbb{R}^{n \times n}$, which is computationally expensive and memory consuming, especially in high dimensional spaces. For example, for ImageNet where $n = 224 \times 224 \times 3$, the transportation

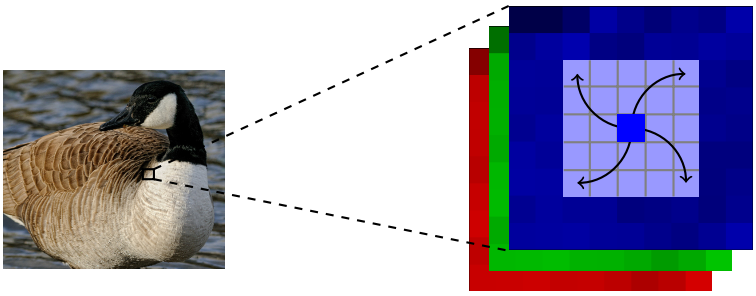


Figure 2.2: An illustration of 5×5 local transportation.

matrix Π has billions of variables and the memory storage (roughly 28.1 GB using single precision floating point numbers) for it exceeds the limit of most GPUs.

To accelerate computation and reduce memory usage, we use the same local transportation technique as introduced by Wong et al. (2019), to impose a *structured sparsity* in Π. Specifically, we only allow moving pixels in a $k \times k$ neighborhood (see Figure 2.2). For images with multiple channels, we only allow transportation within each channel. Adversarial examples generated with these restrictions are still valid under the original Wasserstein threat models.

With local transportation, each pixel can be only transported to at most k^2 possible locations, thus Π is a highly sparse matrix with at most k^2 nonzero entries in each row. Such sparsity reduces the per iteration cost of dual projection from $O(n^2 \log n)$ to $O(nk^2 \log k)$, and reduces that of dual LMO from $O(n^2)$ to $O(nk^2)$, both of which are linear w.r.t. n, treating k (typically much smaller than n) as a constant.

Implementing sparse matrices computation on GPUs is not trivial, as operations on sparse matrices, in general, are not easy to parallelize on GPUs. As such, current deep learning packages (PyTorch, TensorFlow) do not support general sparse matrices well. Nevertheless, for dual projection and dual LMO, we do have simple customized strategies to support the sparse operations by exploiting the sparsity pattern. Notice that each row of Π has at most k^2 nonzero entries. We store Π as a $n \times k^2$ dense matrix, with some possible dummy entries. The advantage is that now Π is a dense matrix. Any row operations on Π (*e.g.* softmin along each row, sorting along each row) can be parallelized easily. The downside, however, is that column operations (*e.g.* summation over each column) might take extra efforts. Nevertheless, this is not a bottleneck of the speed of dual projection and dual LMO, since they only require efficient row operations.

2.4.2 Gradient Normalization

In both PGD and FW, we normalize the gradient in each iteration by its largest absolute value of entries. For PGD, gradient normalization is a standard practice, yielding consistent scale of gradient and easing step size tuning for nonsmooth objectives. For FW, normalization in the linear minimization step (2.9) does not change the minimizer, but it does affect the scale of the entropic regularization γ in (2.13). In this case, normalization keeps the scale of entropic regularization consistent. In addition, gradient normalization leads to more consistent upper bounds on the dual solutions in (2.7) and (2.16).

2.4.3 Hypercube Constraint in Image Domain

For image domain adversarial examples, there is an additional hypercube constraint, *e.g.*, $\mathbf{x}_{adv} \in [0,1]^n$ if pixels are represented using real numbers in $[0,1]$. In practice, we observe that solving problem (2.1) often generates adversarial examples that violate the hypercube constraint, *e.g.*, some pixels of adversarial images exceed 1. Although simply clipping pixels can enforce the hypercube constraint, certain amount of pixel mass is lost during clipping, which leads to undefined Wasserstein distance. This is in sharp contrast to ℓ_p threat models, where clipping is the typical practice that still retains feasibility hence validness of generated adversarial examples.

To address this issue, we develop another specialized quadratic programming solver to project a transportation matrix Π to the intersection of both constraints. Suppose that pixel values of input images are represented by real numbers in $[0,1]^n$. We need to add one additional constraint $\Pi^\top \mathbf{1} \leq \mathbf{1}$, arriving at the following Euclidean projection problem:

$$\begin{aligned} \underset{\Pi \geq 0}{\text{minimize}} \quad & \frac{1}{2}\|\Pi - G\|_{\mathrm{F}}^2 \\ \text{subject to} \quad & \Pi \mathbf{1} = \mathbf{x}, \ \Pi^\top \mathbf{1} \leq \mathbf{1}, \langle \Pi, C \rangle \leq \delta. \end{aligned}$$

We call this problem a *capacity constrained projection*, since the additional constraint essentially specifies the maximum mass that a pixel location can receive. We introduce the partial Lagrangian to derive the following dual problem:

$$\underset{\lambda \geq 0, \boldsymbol{\mu} \geq 0}{\text{maximize}} \ g(\lambda, \boldsymbol{\mu}),$$

where

$$
\begin{aligned}
g(\lambda, \boldsymbol{\mu}) &= \min_{\Pi \geq 0, \Pi \mathbf{1} = \mathbf{x}} \ \frac{1}{2}\|\Pi - G\|_{\mathrm{F}}^2 + \lambda\left(\langle \Pi, C\rangle - \delta\right) + \boldsymbol{\mu}^\top\left(\Pi^\top \mathbf{1} - \mathbf{1}\right) \\
&= \min_{\Pi \geq 0, \Pi \mathbf{1} = \mathbf{x}} \ \frac{1}{2}\|\Pi - G\|_{\mathrm{F}}^2 + \lambda\left(\langle \Pi, C\rangle - \delta\right) + \langle \Pi, \mathbf{1}\boldsymbol{\mu}^\top\rangle - \boldsymbol{\mu}^\top \mathbf{1} \\
&= \min_{\Pi \geq 0, \Pi \mathbf{1} = \mathbf{x}} \ \frac{1}{2}\|\Pi - G + \lambda C + \mathbf{1}\boldsymbol{\mu}^\top\|_{\mathrm{F}}^2 - \langle G, \lambda C + \mathbf{1}\boldsymbol{\mu}^\top\rangle + \frac{1}{2}\|\lambda C + \mathbf{1}\boldsymbol{\mu}^\top\|_{\mathrm{F}}^2 - \lambda\delta - \boldsymbol{\mu}^\top \mathbf{1}
\end{aligned}
$$

We optimize $g(\lambda, \boldsymbol{\mu})$ by alternating maximization on λ and $\boldsymbol{\mu}$ respectively: fixing $\boldsymbol{\mu}$, we maximize λ via bisection method; fixing λ, we maximize $\boldsymbol{\mu}$ via k steps gradient ascent with nesterov acceleration, where k is a hyperparameter. Evaluating the gradient of $\boldsymbol{\mu}$ in bisection method, as well as evaluating the gradient of $\boldsymbol{\mu}$, can be reduced to simplex projections following similar derivations in dual projection (§2.2).

Due to sublinear convergence rate of gradient ascent, it may be slow to obtain a high precision solution. However, empirical evidences show that it is possible to converge to a reasonably accurate solution (*e.g.* satisfying hypercube constraint up to the third digit after the decimal point) with a few hundred iterations of alternating maximization (with k around 10 or 20). Convergence to these modest precision solutions is already sufficient for generating valid Wasserstein adversarial examples (see Appendix C.8).

To the best of our knowledge, the hypercube constraint has not been addressed in previous study of Wasserstein adversarial attacks. For instance, Wong et al. (2019) in their implementation simply applied clipping regardless. This new algorithm, however, is not as efficient as dual projection nor dual LMO. Thus, we recommend using it as a post-processing procedure on $\Pi^\star$ after solving the problem (2.1). Adversarial attacks tend to be weaker after the post-processing. However, it ensures that the generated adversarial images satisfy both the Wasserstein constraint and the hypercube constraint simultaneously hence are genuinely valid. See Appendix C.8 for a case study.

Chapter 3

Evaluation

In this chapter, we present experiments to demonstrate the effectiveness of our algorithms. In §3.1, we present a sanity check on a simple convex problem, where our algorithms are able to converge to high precision solutions. In §3.2, we apply our algorithms to generate adversarial examples on large scale datasets. Extensive experiments demonstrates that our algorithms are able to solve large scale problems more accurately and much faster. In the end, we demonstrate the usage of our algorithms in adversarial training to improve the Wasserstein adversarial robustness of deep models.

3.1 A Synthetic Convex Problem

We start with a simple convex minimization problem as a sanity check of our algorithms. Consider the following (Euclidean) projection of $\mathbf{b}$ onto a Wasserstein ball centered at $\mathbf{a}$ with radius ϵ:

$$\begin{array}{ll} \underset{\mathbf{x}}{\text{minimize}} & \frac{1}{2}\|\mathbf{x}-\mathbf{b}\|^2 \\ \text{subject to} & \mathcal{W}(\mathbf{x},\mathbf{a}) \le \epsilon, \end{array} \tag{3.1}$$

which can be solved approximately by projected Sinkhorn. On the other hand, we can reparametrize (3.1) into the following equivalent minimization problem as in §2.1:

$$\begin{array}{ll} \underset{\Pi}{\text{minimize}} & \frac{1}{2}\|\Pi^\top \mathbf{1}-\mathbf{b}\|^2 \\ \text{subject to} & \Pi \ge 0, \Pi\mathbf{1}=\mathbf{a}, \langle \Pi, C\rangle \le \epsilon. \end{array} \tag{3.2}$$

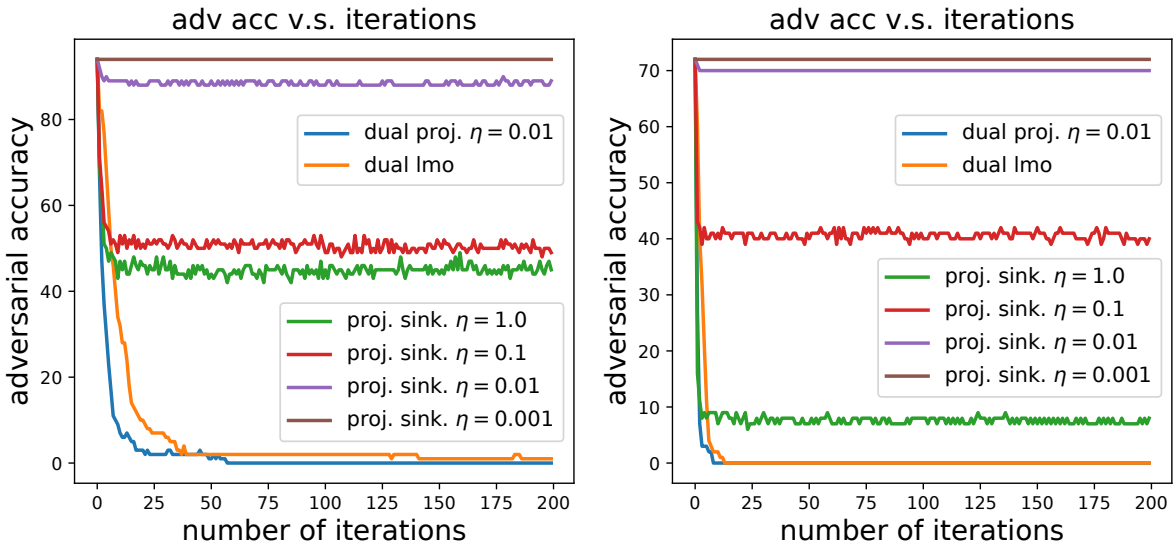
Table 3.1: (Exact) Wasserstein distances $\mathcal{W}(\mathbf{a}, \hat{\mathbf{p}})$ and number of dual iterations in projected Sinkhorn in the first four columns. γ is the entropic regularization constant. Projected Sinkhorn encountered numerical issues for small $\gamma = 5 \cdot 10^{-5}$.

γ	10^{-3}		$2 \cdot 10^{-4}$		10^{-4}		$5 \cdot 10^{-5}$		ours		ground truth
	$\mathcal{W}$	iter	$\mathcal{W}$	iter	$\mathcal{W}$	iter	$\mathcal{W}$	iter	PGD	FW	
$\epsilon = 0.5$	0.267	28	0.402	44	0.437	205	–	–	0.500	0.500	0.500
$\epsilon = 1.0$	0.356	21	0.498	111	0.555	197	–	–	0.797	0.797	0.797

We use projected gradient descent and Frank-Wolfe to solve (3.2) and recover the solution of (3.1) by $(\Pi^\star)^\top \mathbf{1}$.

To compare the above methods, we randomly generate two vectors $\mathbf{a}, \mathbf{b} \in [0, 1]^{400}$ with unit total mass. The initial Wasserstein distance between $\mathbf{a}$ and $\mathbf{b}$ is 0.797. Next, we project $\mathbf{b}$ using projected Sinkhorn onto Wasserstein balls centered at $\mathbf{a}$ for two different radii $\epsilon = 0.5$ and $\epsilon = 1.0$ respectively, by projected Sinkhorn solving (3.1). In comparison, we solve (3.2) iteratively by PGD with dual projection (§2.2) and by Frank-Wolfe with dual LMO (§2.3).

We report in Table 3.1 the number of iterations for projected Sinkhorn to output an approximate projection $\hat{\mathbf{p}}$, and we compute the (exact) Wasserstein distance between $\mathbf{a}$ and $\hat{\mathbf{p}}$ using a linear programming solver for (1.1). In the first row of Table 3.1, we expect $\mathcal{W}(\mathbf{a}, \hat{\mathbf{p}}) = 0.5$, since for an exterior point the exact projection should be at the boundary of the Wasserstein ball. In the second row, we expect $\mathcal{W}(\mathbf{a}, \hat{\mathbf{p}}) = 0.797$, since an exact projection of an interior point should be itself. However, in both cases, the actual Wasserstein distances of projected Sinkhorn are always much smaller than the ground truth. Although $\mathcal{W}(\mathbf{a}, \hat{\mathbf{p}})$ gets closer to the ground truth as γ (the entropic regularization constant) decreases, non-negligible gaps remain. Further decreasing γ may potentially reduce the approximation error, but (a) overly small γ causes numerical issues easily and (b) the number of iterations increases as γ decreases. As we will confirm in §3.2, this approximation error in projected Sinkhorn leads to a substantially suboptimal attack. Their outputs are exact in the first three digits after the decimal point, which serves as a simple sanity check of our algorithms in the convex setting.



(a) CIFAR-10 (b) ImageNet

Figure 3.1: Adversarial accuracy of models w.r.t. different iterations of attacks using $\epsilon = 0.005$. Projected Sinkhorn uses $\gamma = 5 \cdot 10^{-5}$ on CIFAR-10 and $5 \cdot 10^{-6}$ on ImageNet. Dual LMO uses $\gamma = 10^{-3}$ and decay schedule $\frac{2}{t+1}$.

3.2 Large Scale Experiments

Datasets and models We conduct experiments on MNIST (LeCun 1998), CIFAR-10 (Krizhevsky 2009) and ImageNet (Deng et al. 2009). On MNIST and CIFAR-10, we attack two deep networks used by Wong et al. (2019). On ImageNet, we attack a 50-layer residual network (He et al. 2016). ImageNet experiments are run on the first 100 samples in the validation set. See model details in Appendix C.

Choice of cost Throughout the experiment, the cost matrix C is defined as $C_{(i_1,j_1)(i_2,j_2)} = \sqrt{(i_1 - i_2)^2 + (j_1 - j_2)^2}$, namely, the Euclidean distance between pixel indices. We use 5×5 local transportation plan (see §2.4.1) to accelerate computation of all algorithms.

Choice of γ We follow all parameter settings reported by Wong et al. (2019) for projected Sinkhorn, except that we try a few more γ. For dual LMO, we use a fixed $\gamma = 10^{-3}$, which we find to work well across different datasets.

Optimization parameters Stopping criteria of projected Sinkhorn, dual projection and dual LMO, as well as the choice of step sizes of PGD, are deferred to Appendix C. FW

uses a fixed decay schedule $\eta_t = \frac{2}{t+1}$. Step sizes of PGD are tuned in $\{1, 10^{-1}, 10^{-2}, 10^{-3}\}$. Some experiments for different step sizes are presented in §3.2.1.

3.2.1 Convergence Speed of Outer Maximization

Our method depends on a different but equivalent formulation (2.1) that simplifies the constraint. It is reasonble to ask: could the reformulation make the outer maximization in PGD and FW harder? Intuitively, it does not, since the formulation simply embeds a linear transformation before the input layer (summation over columns of Π). To verify this, we plot adversarial accuracy of models w.r.t. number of iterations for different attack algorithms in Figure 3.1 to compare their convergence rate of outer maximization. More thorough results are deferred to Appendix C.4. [1]

We observe that FW with the default decay schedule converges very fast, especially at the initial stage. Meanwhile, PGD with dual projection converges slightly faster than FW with carefully tuned step sizes. In contrast, PGD with projected Sinkhorn barely decreases the accuracy when using small step sizes. The output of projected Sinkhorn is only a feasible point in the Wasserstein ball, rather than an accurate projection, due to the crude approximation. If using small step sizes, projected Sinkhorn brings the iterates of PGD closer to the center of Wasserstein ball in every iteration. Thus, the iterates always stay around the center of Wasserstein ball during the optimization, hence cannot decrease the accuracy. To make progress in optimization, it is required to use aggressively large step sizes (*e.g.* $\eta = 1.0$ and $\eta = 0.1$).

3.2.2 Attack Strength and Dual Convergence Speed

In Table 3.2, we compare (a) strength of different attacks by adversarial accuracy, *i.e.* model accuracy under attacks and (b) the running speed by the average number of dual iterations. We observe that PGD with dual projection attack and FW with dual LMO attack are generally stronger than PGD with projected Sinkhorn, since the latter is only an *approximate* projection hence it does not solve (1.3) adequately. As γ decreases, PGD with projected Sinkhorn gradually becomes stronger due to better approximation, but at a cost of an increasing number of iterations to converge. However, projected Sinkhorn is still weaker than PGD with dual projection and FW with dual LMO, even after tuning

[1] The numbers in Figure 3.1 are meant for comparison of convergence speed of Wasserstein constrained optimization problem. Thus they are shown without the post-processing algorithm discussed in §2.4.3 and may differ slightly from those in Table 3.2.

Table 3.2: Comparison of adversarial accuracy and average number of dual iterations. $\gamma = \frac{1}{1000}$ on MNIST and $\gamma = \frac{1}{3000}$ on CIFAR-10 are the parameters used by Wong et al. (2019). "–" indicates numerical issues during computation.

	method	$\epsilon = 0.1$		$\epsilon = 0.2$		$\epsilon = 0.3$		$\epsilon = 0.4$		$\epsilon = 0.5$	
		acc	iter	acc	iter	acc	iter	acc	iter	acc	iter
MNIST	PGD + Proj. Sink. ($\gamma = 1/1000$)	96.5	92	91.2	88	78.0	85	59.1	82	40.1	80
	PGD + Proj. Sink. ($\gamma = 1/1500$)	95.2	110	82.3	116	58.2	112	–	–	–	–
	PGD + Proj. Sink. ($\gamma = 1/2000$)	–	–	–	–	–	–	–	–	–	–
	PGD + Dual Proj.	63.4	15	13.3	15	1.4	15	0.1	15	0.0	15
	FW + Dual LMO ($\gamma = 10^{-3}$)	67.5	15	16.9	15	2.2	15	0.4	15	0.1	15
	PGD + Dual Proj. (w/o post-processing)	42.6	15	4.2	15	0.4	15	0.0	15	0.0	15
	FW + Dual LMO (w/o post-processing)	48.3	15	6.7	15	1.0	15	0.3	15	0.1	15
	method	$\epsilon = 0.001$		$\epsilon = 0.002$		$\epsilon = 0.003$		$\epsilon = 0.004$		$\epsilon = 0.005$	
		acc	iter	acc	iter	acc	iter	acc	iter	acc	iter
CIFAR-10	PGD + Proj. Sink. ($\gamma = 1/3000$)	93.0	33	91.3	33	89.5	33	87.6	33	85.7	33
	PGD + Proj. Sink. ($\gamma = 1/10000$)	89.9	79	84.5	79	78.3	79	71.9	79	65.6	79
	PGD + Proj. Sink. ($\gamma = 1/20000$)	–	–	–	–	–	–	–	–	–	–
	PGD + Dual Proj.	30.3	15	10.5	15	5.6	15	4.0	15	3.4	15
	FW + Dual LMO ($\gamma = 10^{-3}$)	33.5	15	13.6	15	7.2	15	4.7	15	3.7	15
	PGD + Dual Proj. (w/o post-processing)	25.9	15	6.0	15	1.7	15	0.5	15	0.2	15
	FW + Dual LMO (w/o post-processing)	29.6	15	9.1	15	3.2	15	1.1	15	0.6	15
	method	$\epsilon = 0.001$		$\epsilon = 0.002$		$\epsilon = 0.003$		$\epsilon = 0.004$		$\epsilon = 0.005$	
		acc	iter	acc	iter	acc	iter	acc	iter	acc	iter
ImageNet	PGD + Proj. Sink. ($\gamma = 1/100000$)	68.0	42	61.2	43	59.2	43	55.8	43	52.4	43
	PGD + Proj. Sink. ($\gamma = 1/200000$)	61.2	72	56.5	72	48.3	72	42.9	71	38.1	71
	PGD + Proj. Sink. ($\gamma = 1/1000000$)	–	–	–	–	–	–	–	–	–	–
	PGD + Dual Proj.	9.0	15	9.0	15	8.0	15	8.0	15	7.0	15
	FW + Dual LMO	10.0	15	9.0	15	8.0	15	8.0	15	7.0	15
	PGD + Dual Proj. (w/o post-processing)	0.0	15	0.0	15	0.0	15	0.0	15	0.0	15
	FW + Dual LMO (w/o post-processing)	1.0	15	0.0	15	0.0	15	0.0	15	0.0	15

γ. Unfortunately, further decreasing γ runs into numerical overflow. We notice that PGD with dual projection is often slightly stronger than FW with dual LMO for two possible reasons: the projection step is solved exactly without any approximation error; we use the default decay schedule in FW. Tuning the decay schedule for specific problems might improve the attack strength and convergence speed of FW.

For completeness, we also report the results of dual projection and dual LMO without post-processing in Table 3.2. After post-processing (see §2.4.3), the adversarial accuracy is increased, sometimes by a lot. This is especially the case on MNIST (*e.g.*, $\epsilon = 0.1$) where there are many white pixels, thus it is very easy to violate the hypercube constraint. Note that PGD with projected Sinkhorn might be even weaker than what is indicated by the statistics in Table 3.2, if we post-process its adversarial examples appropriately so that they are genuinely valid. However, we do not have an efficient algorithm for post-processing

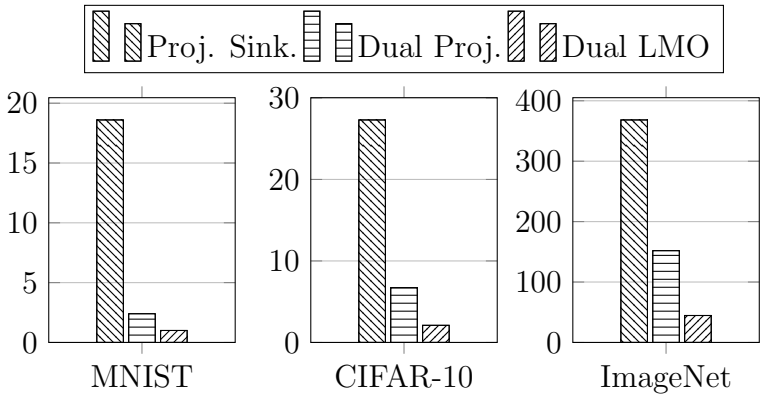


Figure 3.2: Per iteration running time (in milliseconds) of different algorithms measured on a single P100 GPU.

projected Sinkhorn, thus we simply let it ignore the hypercube constraint. Even so, our attacks are still much stronger than it.

Thanks to the bisection method and the tight upper bounds (2.7) and (2.16), dual projection and dual LMO converge very fast to high precision solutions. In practice, they often terminate exactly in 15 iterations due to the consistent scales of the upper bounds (see Appendix C.2 for a discussion). On the other hand, projected Sinkhorn typically requires more dual iterations. Besides convergence speed, we also compare the real running time of a single dual iteration of all three methods in Figure 3.2. On MNIST and CIFAR-10, dual projection is $5 \sim 7$ times faster than projected Sinkhorn; while on ImageNet, dual projection is roughly twice faster than projected Sinkhorn. Meanwhile, dual LMO is $2 \sim 3$ times faster than dual projection due to the absence of the extra logarithm factor. See Appendix C.7 for more details.

3.2.3 Entropic Regularization Reflects Shapes in Images

Wong et al. (2019) noted that Wasserstein perturbations reflect shapes in original images. Instead, we argue that it is the large entropic regularization that causes the phenomenon. We visualize adversarial examples and perturbations generated by different attacks in Figure 3.3 and Figure 3.4. Perturbations generated by PGD with dual projection and FW with dual LMO using small γ tend to be very sparse, *i.e.*, only moving a few pixels in the images. In comparison, we gradually increase the entropic regularization in dual LMO and eventually are able to reproduce the shape reflection phenomenon observed by Wong

et al. (2019). The fact that projected Sinkhorn generates adversarial perturbations reflecting shapes in clean images could be another evidence that the entropic regularization is too large. Notice that large entropic regularization causes large approximation error, thus potentially requires larger ϵ in order to successfully generate adversarial examples.

A large entropic regularization term in the optimization objective (of projected Sinkhorn and dual LMO) encourages the transportation to be uniform, thus each pixel tends to spread its mass evenly to its nearby pixels. In the region where pixel intensities do not change much, the transportations cancel out; while in the region where pixel intensities change drastically (*e.g.*, edges in an image), pixel mass flows from the high pixel region to low pixel region. As a result, it reflects the edges in the original image.

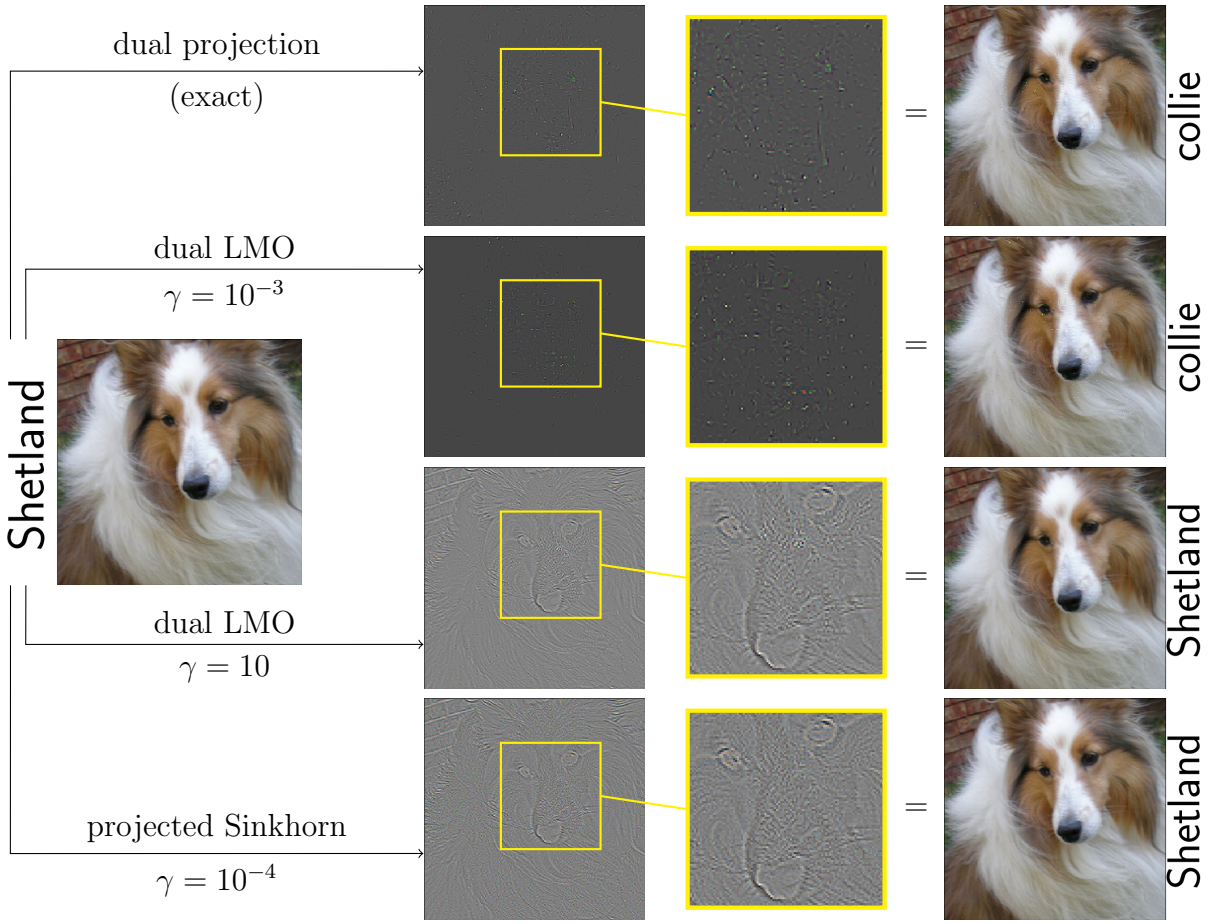


Figure 3.3: Wasserstein adversarial examples ($\epsilon = 0.005$) generated by different algorithms on ImageNet. Perturbations are normalized to $[0, 1]$ for visualization. Dog faces can be observed after zooming in.

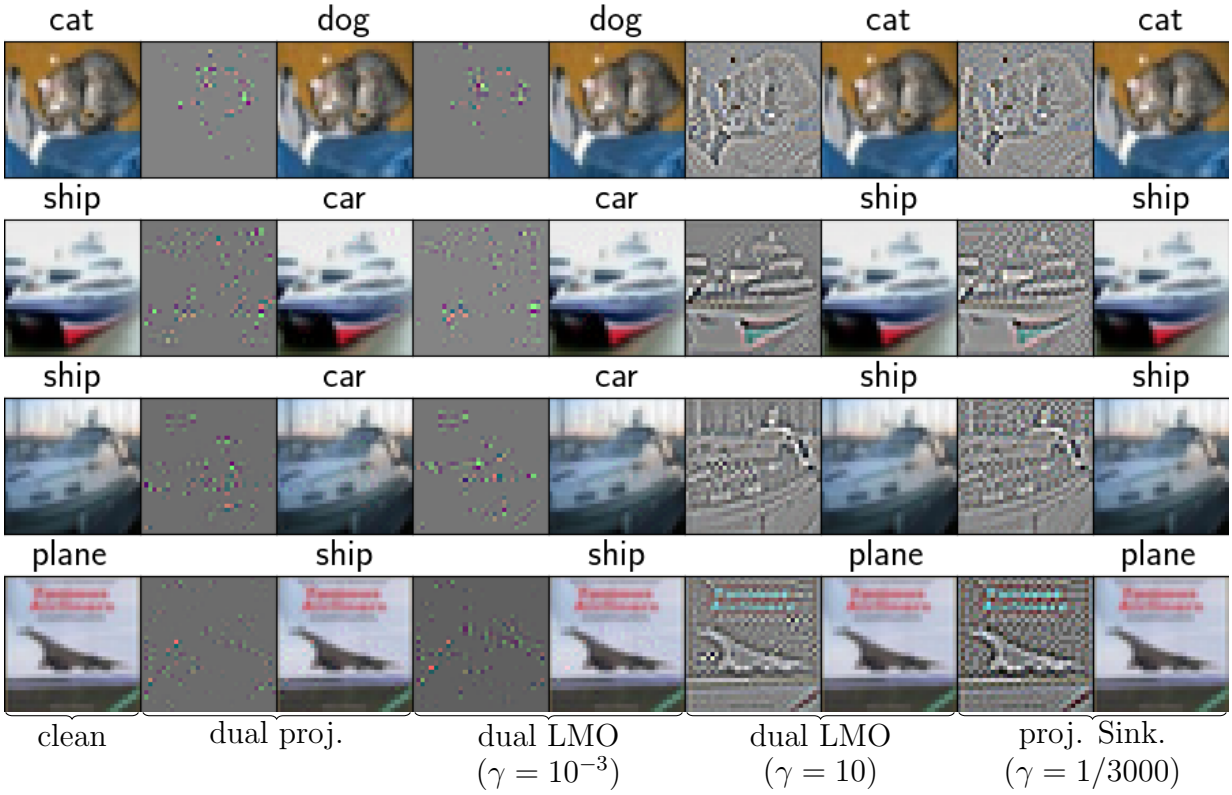


Figure 3.4: Wasserstein adversarial examples ($\epsilon = 0.002$) generated by different algorithms on CIFAR-10. Perturbations reflect shapes in images only when using large entropic regularization (the 6th and 8th columns).

3.2.4 Improved Adversarial Training

In this section, we show how to use our stronger attacks in adversarial training (Madry et al. 2018) to improve the robustness of deep models. Adversarial training is one of the most effective empirical defense. Instead of minimizing empirical loss over the training data, adversarial training minimizes the worst case loss:

$$\underset{\boldsymbol{\theta}}{\text{minimize}} \ \mathbf{E}_{(\mathbf{x},y)\sim p_{\text{data}}} \left[\max_{\mathcal{W}(\mathbf{x},\mathbf{z})\leq\delta} \ell(f_{\boldsymbol{\theta}}(\mathbf{z}), y) \right], \tag{3.3}$$

where for each $\mathbf{x}$ the loss is maximized over a small neighbourhood around $\mathbf{x}$. During training, we use projected gradient ascent to approximate the inner maximization, *i.e.* generating an adversarial example $\mathbf{x}_{adv}$, and then minimize training loss at $\mathbf{x}_{adv}$ Due to its simplicity and effectiveness, adversarial training is particularly popular as an empirical defense. In fact, Rice et al. (2020) show that it is the strongest defense without additional unlabeled data.

Since we have developed stronger attacks, it is natural that applying them in the inner maximization step of Equation (3.3) will improve the robustness of adversarially trained models. Indeed, models adversarially trained by our stronger attack have much higher adversarial accuracy compared with models trained by projected Sinkhorn. We apply FW with dual LMO in adversarial training due to its fast convergence speed and fast per iteration running speed. On MNIST, we produce a robust model with 59.7% accuracy against all three attacks with perturbation budget $\epsilon = 0.5$, compared with 0.6% using projected Sinkhorn. On CIFAR-10, we produce a robust model with 56.8% accuracy against all three attacks with perturbation budget $\epsilon = 0.005$, compared with 41.2% using projected Sinkhorn. We present a thorough evaluation of adversarially trained models in the following two sections,

MNIST

We adversarially train a robust model using Frank-Wolfe with dual LMO and a fixed perturbation budget $\epsilon = 0.3$. The inner maximization is approximated with 40 iterations of Frank-Wolfe.[2] This model achieves 95.83% clean accuracy. The adversarial accuracy of this model is shown in Table 3.3.

For comparison, we also present results on attacking an adversarially trained model by PGD with projected Sinkhorn. We use a pretrained model released by Wong et al. (2019),

[2]We also have tried larger perturbation budgets $\epsilon = 0.4$ and $\epsilon = 0.5$, but the training collapses.

Table 3.3: MNIST model adversarially trained by Frank-Wolfe with dual LMO ($\epsilon = 0.3$).

method	$\epsilon = 0.1$	$\epsilon = 0.2$	$\epsilon = 0.3$	$\epsilon = 0.4$	$\epsilon = 0.5$
PGD + Proj. Sink. ($\gamma = 1/1000$)	94.9	94.0	93.0	91.9	90.5
PGD + Proj. Sink. ($\gamma = 1/1500$)	94.5	93.2	91.5	89.3	86.8
PGD + Proj. Sink. ($\gamma = 1/2000$)	–	–	–	–	–
PGD + Dual Proj.	92.6	87.8	80.2	70.7	59.7
FW + Dual LMO	92.5	88.1	82.1	75.3	66.8
PGD + Dual Proj. (w/o post-processing)	91.1	82.9	71.3	58.4	44.6
FW + Dual LMO (w/o post-processing)	91.1	83.9	73.9	63.0	50.9

Table 3.4: MNIST model adversarially trained by PGD with projected Sinkhorn.

method	$\epsilon = 0.1$	$\epsilon = 0.2$	$\epsilon = 0.3$	$\epsilon = 0.4$	$\epsilon = 0.5$
PGD + Proj. Sink. ($\gamma = 1/1000$)	95.0	92.4	90.5	88.5	86.5
PGD + Proj. Sink. ($\gamma = 1/1500$)	93.8	90.0	82.7	85.2	90.5
PGD + Proj. Sink. ($\gamma = 1/2000$)	–	–	–	–	–
PGD + Dual Proj.	1.1	0.8	0.6	0.6	0.6
FW + Dual LMO	4.8	21.6	34.5	39.2	39.6
PGD + Dual Proj. (w/o post-processing)	1.0	0.4	0.3	0.3	0.3
FW + Dual LMO (w/o post-processing)	0.8	0.3	0.2	0.1	0.0

which is adversarially trained using an adaptive perturbation budget $\epsilon \in [0.1, 2.1]$. This model achieves 97.28% clean accuracy.

We notice that the model adversarially trained by PGD with projected Sinkhorn seems to overfit to the same attack. Compared with the standard trained model in Table 3.2, the adversarially trained model has higher adversarial accuracy under projected Sinkhorn, but has lower adversarial accuracy under our stronger attacks.

In Table 3.4, the post-processing algorithm does not work quite well with Frank-Wolfe. Normally, increasing the perturbation budget ϵ should decrease the adversarial accuracy. Indeed, if we do not post-process the output, the adversarial accuracy decreases monotonically for Frank-Wolfe. However, after post-processing, the adversarial accuracy increases a lot, and even increases as the the perturbation budget increases. For this model, our post-processing algorithm does not seems to be "compatible" with Frank-Wolfe. Doing a better job on optimizing the Wasserstein constrained problem ignoring the hypercube constraint does not necessarily give a good solution to the problem with hypercube constraint.

Table 3.5: CIFAR-10 model adversarially trained by FW with dual LMO ($\epsilon = 0.005$)

method	$\epsilon = 0.001$	$\epsilon = 0.002$	$\epsilon = 0.003$	$\epsilon = 0.004$	$\epsilon = 0.005$
PGD + Proj. Sink. ($\gamma = 1/3000$)	82.4	82.3	82.1	81.9	81.8
PGD + Proj. Sink. ($\gamma = 1/10000$)	82.0	81.5	81.0	80.6	80.1
PGD + Proj. Sink. ($\gamma = 1/20000$)	–	–	–	–	–
PGD + Dual Proj.	76.8	71.8	67.0	62.0	56.8
FW + Dual LMO	77.4	73.7	70.2	66.8	62.6
PGD + Dual Proj. (w/o post-processing)	76.5	71.3	66.4	61.4	56.2
FW + Dual LMO (w/o post-processing)	77.2	73.7	70.4	67.3	63.9

Table 3.6: CIFAR-10 model adversarially trained by PGD with projected Sinkhorn.

method	$\epsilon = 0.001$	$\epsilon = 0.002$	$\epsilon = 0.003$	$\epsilon = 0.004$	$\epsilon = 0.005$
PGD + Proj. Sink. ($\gamma = 1/3000$)	81.7	81.6	81.6	81.6	81.5
PGD + Proj. Sink. ($\gamma = 1/10000$)	81.6	81.4	81.3	81.2	81.0
PGD + Proj. Sink. ($\gamma = 1/20000$)	–	–	–	–	–
PGD + Dual Proj.	72.5	64.4	56.3	48.8	42.2
FW + Dual LMO	72.4	64.0	55.5	47.8	41.2
PGD + Dual Proj. (w/o post-processing)	72.0	63.3	54.9	47.2	40.6
FW + Dual LMO (w/o post-processing)	71.7	62.2	52.9	44.7	37.5

CIFAR-10

We adversarially train a robust model using Frank-Wolfe with dual LMO and a fixed perturbation budget $\epsilon = 0.005$. The inner maximization is approximated with at most 30 iterations of Frank-Wolfe. This model achieves 82.57% clean accuracy. The adversarial accuracy of this model is shown in Table 3.5.

For comparison, we also present results on attacking an adversarially trained model by PGD with projected Sinkhorn. We use a pretrained model released by Wong et al. (2019), which is adversarially trained using an adaptive perturbation budget $\epsilon \in [0.01, 0.38]$. This model achieves 81.68% clean accuracy.

Chapter 4

Conclusion

In this thesis, we have demonstrated that the previous Wasserstein adversarial attack based on *approximate* projection is suboptimal due to inaccurate projection. To generate stronger Wasserstein adversarial attacks, we introduce two *faster* and more *accurate* algorithms for Wasserstein constrained optimization. Each algorithm has its own advantage thus they complement each other nicely: PGD with dual projection employs exact projection and generates the strongest attack. On the other hand, with minimal entropic smoothing, FW with dual LMO is extremely fast in terms of both outer maximization and linear minimization step without much tuning of hyperparameters. Extensive experiments confirm the effectiveness of our algorithms in two ways: (a) properly evaluating Wasserstein adversarial robustness and (b) improving robustness through adversarial training.

A key ingredient of our methods is that we design partial Lagrangian such that (a) minimizing the Lagrangian is easy (almost closed form) and (b) solving the dual is easy (univariate), which allows us to develop fast algorithms with high precision. While we demonstrate it is possible to do so for Wasserstein constraint by exploring the structure of the problems, extending this method for additional constraints seems to be nontrivial, as we already see in §2.4.3 for example. Thus, developing algorithms for variants of Wasserstein distance (*e.g.* Korman et al. 2015) might need further careful design to balance the solver precision and computational efficiency.

Finally, our algorithms impose minimal assumptions on the cost matrix in Wasserstein distance, thus they can be directly applied to other applications involving Wasserstein constrained optimization problems on discrete domains (*e.g.*, Wasserstein distributionally robust optimization (Kuhn et al. 2019)).

References

[1] Rima Alaifari, Giovanni S. Alberti, and Tandri Gauksson. "ADef: An Iterative Algorithm to Construct Adversarial Deformations". In: *International Conference on Learning Representations*. 2019.

[2] Martin Arjovsky, Soumith Chintala, and Léon Bottou. "Wasserstein Generative Adversarial Networks". In: *Proceedings of the 34th International Conference on Machine Learning*. 2017, pp. 214–223.

[3] Anish Athalye, Nicholas Carlini, and David Wagner. "Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples". In: *Proceedings of the 35th International Conference on Machine Learning*. 2018, pp. 274–283.

[4] Anish Athalye, Logan Engstrom, Andrew Ilyas, and Kevin Kwok. "Synthesizing Robust Adversarial Examples". In: *Proceedings of the 35th International Conference on Machine Learning*. 2018, pp. 284–293.

[5] Mislav Balunovic, Maximilian Baader, Gagandeep Singh, Timon Gehr, and Martin Vechev. "Certifying Geometric Robustness of Neural Networks". In: *Advances in Neural Information Processing Systems 32*. 2019, pp. 15313–15323.

[6] James P Boyle and Richard L Dykstra. "A Method for Finding Projections onto the Intersection of Convex Sets in Hilbert Spaces". In: *Advances in order restricted statistical inference*. 1986, pp. 28–47.

[7] Nicholas Carlini and David Wagner. "Towards Evaluating the Robustness of Neural Networks". In: *2017 IEEE Symposium on Security and Privacy (SP)*. 2017, pp. 39–57.

[8] Jinghui Chen, Dongruo Zhou, Jinfeng Yi, and Quanquan Gu. "A Frank-Wolfe Framework for Efficient and Effective Adversarial Attacks". In: *AAAI*. 2020.

[9] Jeremy Cohen, Elan Rosenfeld, and Zico Kolter. "Certified Adversarial Robustness via Randomized Smoothing". In: *Proceedings of the 36th International Conference on Machine Learning*. 2019, pp. 1310–1320.

[10] R. Cominetti and J. San Martín. "Asymptotic Analysis of the Exponential Penalty Trajectory in Linear Programming". In: *Mathematical Programming* 67 (1994), pp. 169–187.

[11] Marco Cuturi. "Sinkhorn Distances: Lightspeed Computation of Optimal Transport". In: *Advances in Neural Information Processing Systems 26*. 2013, pp. 2292–2300.

[12] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Fei-Fei Li. "ImageNet: A Large-scale Hierarchical Image Database". In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 2009, pp. 248–255.

[13] Frank Deutsch and Hein Hundal. "The Rate of Convergence of Dykstra's Cyclic Projections Algorithm: The Polyhedral Case". In: *Numerical Functional Analysis and Optimization* 15 (1994), pp. 537–565.

[14] John Duchi, Shai Shalev-Shwartz, Yoram Singer, and Tushar Chandra. "Efficient Projections onto the ℓ_1-ball for Learning in High Dimensions". In: *Proceedings of the 25th international conference on Machine learning*. 2008, pp. 272–279.

[15] Richard L Dykstra. "An Algorithm for Restricted Least Squares Regression". In: *Journal of the American Statistical Association* 78 (1983), pp. 837–842.

[16] Logan Engstrom, Brandon Tran, Dimitris Tsipras, Ludwig Schmidt, and Aleksander Madry. "Exploring the Landscape of Spatial Robustness". In: *Proceedings of the 36th International Conference on Machine Learning*. 2019, pp. 1802–1811.

[17] Marguerite Frank and Philip Wolfe. "An Algorithm for Quadratic Programming". In: *Naval Research Logistics (NRL)* 3 (1956), pp. 95–110.

[18] Ian Goodfellow, Patrick McDaniel, and Nicolas Papernot. "Making Machine Learning Robust Against Adversarial Inputs". In: *Communications of the ACM* 61 (2018), pp. 56–66.

[19] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. "Explaining and harnessing adversarial examples". In: *International Conference on Learning Representation*. 2015.

[20] Sven Gowal, Krishnamurthy (Dj) Dvijotham, Robert Stanforth, Rudy Bunel, Chongli Qin, Jonathan Uesato, Relja Arandjelovic, Timothy Mann, and Pushmeet Kohli. "Scalable Verified Training for Provably Robust Image Classification". In: *The IEEE International Conference on Computer Vision (ICCV)*. 2019.

[21] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep Residual Learning for Image Recognition". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[22] Can Kanbak, Seyed-Mohsen Moosavi-Dezfooli, and Pascal Frossard. "Geometric Robustness of Deep Networks: Analysis and Improvement". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[23] Jonathan Korman and Robert McCann. "Optimal transportation with capacity constraints". In: *Transactions of the American Mathematical Society* 367 (2015), pp. 1501–1521.

[24] Alex Krizhevsky. "Learning Multiple Layers of Features from Tiny Images". Technical Report. 2009.

[25] Daniel Kuhn, Peyman Mohajerin Esfahani, Viet Anh Nguyen, and Soroosh Shafieezadeh-Abadeh. "Wasserstein distributionally robust optimization: Theory and applications in machine learning". In: *Operations Research & Management Science in the Age of Analytics*. 2019, pp. 130–166.

[26] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. "Adversarial Machine Learning at Scale". In: *International Conference on Learning Representations*. 2017.

[27] Cassidy Laidlaw and Soheil Feizi. "Functional Adversarial Attacks". In: *Advances in Neural Information Processing Systems 32*. 2019, pp. 10408–10418.

[28] Yann LeCun. *The MNIST Dataset*. 1998.

[29] Alexander Levine and Soheil Feizi. "Wasserstein Smoothing: Certified Robustness against Wasserstein Adversarial Attacks". In: *The 23rd International Conference on Artificial Intelligence and Statistics*. 2020.

[30] Juncheng Li, Frank Schmidt, and Zico Kolter. "Adversarial camera stickers: A physical camera-based attack on deep learning systems". In: *Proceedings of the 36th International Conference on Machine Learning*. 2019, pp. 3896–3904.

[31] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. "Towards Deep Learning Models Resistant to Adversarial Attacks". In: *International Conference on Learning Representations*. 2018.

[32] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. "DeepFool: A Simple and Accurate Method to Fool Deep Neural Networks". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[33] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. "Distillation as a defense to adversarial perturbations against deep neural networks". In: *2016 IEEE Symposium on Security and Privacy (SP)*. 2016, pp. 582–597.

[34] Gabriel Peyré and Marco Cuturi. *Computational Optimal Transport*. Foundations and Trends® in Machine Learning, 2019.

[35] Aditi Raghunathan, Jacob Steinhardt, and Percy S Liang. "Semidefinite relaxations for certifying robustness to adversarial examples". In: *Advances in Neural Information Processing Systems 31*. 2018, pp. 10877–10887.

[36] Leslie Rice, Eric Wong, and J. Zico Kolter. "Overfitting in Adversarially Robust Deep Learning". In: *Proceedings of the 37th International Conference on Machine Learning*. 2020.

[37] Yossi Rubner, Leonidas J Guibas, and Carlo Tomasi. "The earth mover's distance, multi-dimensional scaling, and color-based image retrieval". In: *Proceedings of the ARPA image understanding workshop*. 1997, pp. 661–668.

[38] Richard Sinkhorn. "Diagonal equivalence to matrices with prescribed row and column sums". In: *The American Mathematical Monthly* 74 (1967), pp. 402–405.

[39] Arun Sai Suggala, Adarsh Prasad, Vaishnavh Nagarajan, and Pradeep Ravikumar. "Revisiting Adversarial Risk". In: *The 22nd International Conference on Artificial Intelligence and Statistics*. 2019, pp. 2331–2339.

[40] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. "Intriguing properties of neural networks". In: *International Conference on Learning Representations*. 2014.

[41] Vincent Tjeng, Kai Y. Xiao, and Russ Tedrake. "Evaluating Robustness of Neural Networks with Mixed Integer Programming". In: *International Conference on Learning Representations*. 2019.

[42] Florian Tramer and Dan Boneh. "Adversarial Training and Robustness for Multiple Perturbations". In: *Advances in Neural Information Processing Systems 32*. 2019, pp. 5858–5868.

[43] Z. Wang and A. C. Bovik. "Mean squared error: Love it or leave it? A new look at Signal Fidelity Measures". In: *IEEE Signal Processing Magazine* 26 (2009), pp. 98–117.

[44] Jonathan Weed. "An explicit analysis of the entropic penalty in linear programming". In: *Proceedings of the 31st Conference On Learning Theory*. 2018, pp. 1841–1855.

[45] Eric Wong and Zico Kolter. "Provable Defenses against Adversarial Examples via the Convex Outer Adversarial Polytope". In: *Proceedings of the 35th International Conference on Machine Learning*. 2018, pp. 5286–5295.

[46] Eric Wong, Frank Schmidt, and Zico Kolter. "Wasserstein Adversarial Examples via Projected Sinkhorn Iterations". In: *Proceedings of the 36th International Conference on Machine Learning*. 2019, pp. 6808–6817.

[47] Kaiwen Wu, Allen Houze Wang, and Yaoliang Yu. "Stronger and Faster Wasserstein Adversarial Attacks". In: *Proceedings of the 37th International Conference on Machine Learning*. 2020.

[48] Chaowei Xiao, Jun-Yan Zhu, Bo Li, Warren He, Mingyan Liu, and Dawn Song. "Spatially Transformed Adversarial Examples". In: *International Conference on Learning Representations*. 2018.

[49] Yaoliang Yu, Xinhua Zhang, and Dale Schuurmans. "Generalized Conditional Gradient for Sparse Estimation". In: *Journal of Machine Learning Research* 18 (2017), pp. 1–46.

APPENDICES

Appendix A

Further Analysis and Examples

A.1 Projected Sinkhorn

A.1.1 Analysis of Approximation Error in Projected Sinkhorn

To ensure small approximation error in (1.4), the scale of entropic regularization term should be at least much smaller than the quadratic term:

$$\gamma \sum_{i=1}^{n} \sum_{j=1}^{n} \Pi_{ij} \log \Pi_{ij} \ll \frac{1}{2} \|\mathbf{w} - \mathbf{z}\|_2^2 .$$

Otherwise, the objective (1.4) is dominated by the entropic regularization. However, in practice, it is not always guaranteed, especially when $\mathbf{w}$ is an interior point of the constraint in (1.4).

Consider an simple example where $\mathbf{w} = \mathbf{x} = (\frac{1}{n}, \frac{1}{n}, \cdots \frac{1}{n})^\top$. In that case, the quadratic term $\frac{1}{2}\|\mathbf{x} - \mathbf{z}\|_2^2$ should be as small as zero, since we can let $\mathbf{z} = \mathbf{x}$. However, if $\mathbf{z} = \mathbf{w} = \mathbf{x}$, then Π could be a diagonal matrix $\operatorname{diag}\left(\frac{1}{n}, \frac{1}{n}, \cdots \frac{1}{n}\right)$ (or more generally, $\frac{1}{n}P$, where P is a permutation matrix). Thus, the entropic term becomes

$$\sum_{i=1}^{n} \sum_{j=1}^{n} \Pi_{ij} \log \Pi_{ij} = -\log n, \tag{A.1}$$

reaching its *maximum*. The entropic regularization somewhat conflicts with the the quadratic term. Notice that the scale of (A.1) is much larger than $\frac{1}{2}\|\mathbf{w} - \mathbf{x}\|_2^2$ (which is supposed to

be very close to zero), especially when the dimension n is large. Thus, the objective (1.4) may be dominated by the entropic regularization and solving the projection step accurately requires very small γ.

We make two additional remarks. First, notice that the scale of (A.1) increases as n grows, which requires smaller γ to balance the quadratic term and entropic regularization. This gives the intuition that projected Sinkhorn needs smaller γ in higher dimensional spaces, which is observed in experiments.

Second, the key aspect of the above argument is that $\mathbf{w}$ is relatively close to $\mathbf{x}$, *e.g.* $\mathcal{W}(\mathbf{w}, \mathbf{x}) \leq \epsilon$, such that the quadratic term is so small hence dominated by the entropic regularization. In the case where $\mathbf{w}$ is very far away from $\mathbf{x}$, this argument does not hold anymore. We believe this explains why large step sizes strengthen the attack when using PGD with projected Sinkhorn in experiments. However, PGD with large step sizes tend to be unstable and may not converge to a good solution.

A.1.2 Toy Experiment in Table 3.1

Entries of $\mathbf{a}$ and $\mathbf{b}$ are sampled from a uniform distribution in $[0, 1]^{400}$ independently. After sampling, both vectors are normalized to ensure that the pixel mass summations are exactly 1. We reshape $\mathbf{a}$ and $\mathbf{b}$ to $\mathbb{R}^{20\times 20}$ and view them as images in order to use the procedure of Wong et al. (2019). The cost matrix is induced by Euclidean norm between pixel indices with 5×5 local transportation plan.

For PGD, we use step size 0.05. For Frank-Wolfe, we use $\gamma = 10^{-3}$ and the default decay schedule $\frac{2}{t+1}$. We let both algorithms run for sufficiently many iterations in order to converge to high precision solutions.

A.2 Stopping Criterion for Bisection Method

When the derivative of the dual objective approaches zero, *i.e.*, $\langle \Pi, C \rangle - \delta \approx 0$, the comparison between $\langle \Pi, C \rangle - \delta$ and 0 is getting numerically unstable. Thus, we recommend stopping the bisection method when either the derivative is close to zero, or the gap between the lower bound l and the upper bound u is relatively small.

We recommend using an upper bound u to recover the coupling Π. Since an upper bound u always has a negative derivative, thus the transportation cost constraint $\langle \Pi, C \rangle \leq \delta$ is always satisfied.

We highlight that the bisection method converges very fast in practice, since it shrinks the interval by a factor of 2 in every iteration. Thus, it determines the next 3 digits of $\lambda^\star$ after the decimal point after every 10 iterations.

For a concrete stopping criterion in our experiment, please refer to Appendix C.2.

A.3 Toy Experiment for Dykstra's Algorithm

We randomly sample a vector $\mathbf{x} \in \mathbb{R}^{100}$ and a coupling $\Pi \in \mathbb{R}^{100\times 100}$ (both from a uniform distribution in a hypercube). We normalize $\mathbf{x}$ and Π. We then project Π to $\mathcal{C}_s \cap \mathcal{C}_h$. We set $\delta = 1$ and the cost matrix C is the same as the one in §3.2 (we reshape $\mathbf{x}$ into a 10×10 matrix and view it as an image). The convergence plots are shown in Figure 2.1. Dykstra's algorithm does converge, but at a slow rate.

A.4 Failure of Dual Linear Minimization without Entropic Regularization

This section presents a simple example to demonstrate the failure of dual LMO without adding entropic regularization.

Let $\delta = 0.5$, $\mathbf{x} = (1, 0)^\top$. Let

$$G = \begin{pmatrix} 1 & -1 \\ 0 & 0 \end{pmatrix}, \quad C = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}.$$

Let

$$\Pi = \begin{pmatrix} \Pi_{11} & \Pi_{12} \\ \Pi_{21} & \Pi_{22} \end{pmatrix}.$$

Then the primal linear program is

$$\begin{aligned} &\underset{\Pi_{11},\Pi_{12}\geq 0}{\text{minimize}} \quad \Pi_{11} - \Pi_{12} \\ &\text{subject to} \quad \Pi_{11} + \Pi_{12} = 1, \Pi_{12} \leq 0.5 \end{aligned}$$

It is easy to check that the solution is

$$\Pi^\star = \begin{pmatrix} 0.5 & 0.5 \\ 0 & 0 \end{pmatrix}.$$

The dual linear program is

$$\underset{\lambda\geq 0}{\text{maximize}} -\frac{1}{2}\lambda + \min\{1, -1 + \lambda\}$$

It is easy to see that the dual problem has a unique solution $\lambda^\star = 2$. Now we try to use the following condition to recover the primal solution:

$$\Pi^\star \in \underset{\Pi \geq 0, \Pi \mathbf{1} = \mathbf{x}}{\operatorname{argmin}} \ \langle \Pi, G + \lambda^\star C \rangle - \lambda^\star \delta, \tag{A.2}$$

which is equivalent to

$$\Pi^\star \in \underset{\Pi \geq 0, \Pi \mathbf{1} = \mathbf{x}}{\operatorname{argmin}} \ \left\langle \Pi, \begin{pmatrix} 1 & 1 \\ 2 & 0 \end{pmatrix} \right\rangle. \tag{A.3}$$

But it turns out that any Π of the form

$$\Pi(\alpha) = \begin{pmatrix} \alpha & 1-\alpha \\ 0 & 0 \end{pmatrix},$$

where $0 \leq \alpha \leq 1$, is a minimizer. By varying α, $\Pi(\alpha)$ can be suboptimal ($\alpha = 0$), optimal ($\alpha = 0.5$) or even infeasible ($\alpha = 1$). Thus, $\Pi^\star$ cannot be recovered by only considering the stationary condition.

Of course it is possible to combine (A.2) with other KKT conditions (specifically, complementary slackness and primal feasibility) to obtain one of the primal solutions. Particularly, in the above example, (A.2) along with the complementary slackness determines the unique primal solution $\Pi^\star$. However, there are still two issues. The first issue is that in more general cases, doing so requires solving a linear system whose variables are from a subset of Π, which could be GPU unfriendly. More critically, the above solution is numerically unstable. Suppose that there is a slight numerical inaccuracy due to floating point precision, such that (A.3) becomes

$$\Pi^\star \in \underset{\Pi \geq 0, \Pi \mathbf{1} = \mathbf{x}}{\operatorname{argmin}} \ \left\langle \Pi, \begin{pmatrix} 1+\xi & 1-\xi \\ 2 & 0 \end{pmatrix} \right\rangle. \tag{A.4}$$

for some small constant $\xi > 0$. Now solving (A.4) gives

$$\Pi(1) = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix},$$

which is infeasible.

Appendix B

Proofs

This section presents all proofs in the paper. Recall that without loss of generality we assume that all entries in the cost matrix C are nonnegative and only diagonal entries of C are zeros. All proofs below assume it implicitly.

B.1 Dual Projection

Proposition 1 *The dual of* (2.2) *is*

$$\underset{\lambda \geq 0}{\text{maximize}} \;\; g(\lambda), \tag{2.3}$$

where

$$g(\lambda) = \min_{\Pi \mathbf{1} = \mathbf{x}, \Pi \geq 0} \; \tfrac{1}{2} \|\Pi - G\|_{\mathrm{F}}^2 + \lambda \left(\langle \Pi, C \rangle - \delta \right). \tag{2.4}$$

In addition, the derivative of $g(\lambda)$ *at a point* $\lambda = \tilde{\lambda}$ *is*

$$g'(\tilde{\lambda}) = \langle \tilde{\Pi}, C \rangle - \delta, \tag{2.5}$$

where

$$\tilde{\Pi} = \underset{\Pi \mathbf{1} = \mathbf{x}, \Pi \geq 0}{\operatorname{argmin}} \; \|\Pi - G + \tilde{\lambda} C\|_{\mathrm{F}}^2. \tag{2.6}$$

Both $g(\lambda)$ *and* $g'(\lambda)$ *can be evaluated in* $O(n^2 \log n)$ *time deterministically for any given* λ.

Proof: Introducing the Lagrange multiplier $\lambda \geq 0$ for the constraint $\langle \Pi, C \rangle \leq \delta$, we arrive at the following dual problem

$$\underset{\lambda \geq 0}{\text{maximize}} \;\; g(\lambda),$$

where

$$g(\lambda) = \min_{\Pi \mathbf{1} = \mathbf{x}, \Pi \geq 0} \;\; \frac{1}{2} \|\Pi - G\|_{\mathrm{F}}^2 + \lambda \left(\langle \Pi, C \rangle - \delta \right).$$

We complete the square in the inner problem, which leads to

$$g(\lambda) = \min_{\Pi \mathbf{1} = \mathbf{x}, \Pi \geq 0} \;\; \frac{1}{2} \|\Pi - (G - \lambda C)\|_{\mathrm{F}}^2 - \frac{1}{2} \lambda^2 \|C\|_{\mathrm{F}}^2 + \lambda \langle G, C \rangle - \lambda \delta.$$

Notice that the constraint in the minimization is independent for each row of Π. Thus, it can be reduced to a simplex projection for each row of $G - \lambda C$, which can be solved in $O(n^2 \log n)$ time.

By Danskin's theorem, g is differentiable and the derivative is

$$g'(\tilde{\lambda}) = \langle \tilde{\Pi}, C \rangle - \delta,$$

given the solution $\tilde{\Pi}$ to the minimization problem. ■

Before proving Proposition 2, we first prove Lemma 1, which characterizes the solution of simplex projection in a special case. Intuitively, the projection of a vector to a simplex is very sparse if one of its entries is much larger than the others.

Lemma 1 *Consider the following projection of a vector* $\mathbf{v}$ *to a simplex:*

$$\begin{aligned} \underset{\mathbf{w} \in \mathbb{R}^n}{\text{minimize}} \quad & \|\mathbf{w} - \mathbf{v}\|_2^2 \\ \text{subject to} \quad & \sum_{i=1}^n w_i = z, \;\; w_i \geq 0, \end{aligned}$$

where $z > 0$. *Suppose that there exists* i *such that* $v_i \geq v_j + z$ *for all* $j \neq i$. *Then the solution is* $\mathbf{w}^\star = (0, \cdots 0, z, 0, \cdots, 0)^\top$, *where the only nonzero entry is* $w_i^\star = z$.

Proof: A careful analysis of the simplex projection algorithm (Duchi et al. 2008) would give a proof. Here, we give an alternative simple proof that does not rely on the algorithm. Assume to the contrary that there exists $j \neq i$ such that $w_j^\star > 0$ hence also $w_i^\star < z$. We construct another feasible point $\hat{\mathbf{w}}$ by

$$\begin{aligned}\hat{w}_i &= w_i^\star + w_j^\star \\ \hat{w}_j &= 0 \\ \hat{w}_k &= w_k^\star \quad \forall k \neq i, k \neq j.\end{aligned}$$

Comparing the objective value of $\mathbf{w}^\star$ and $\hat{\mathbf{w}}$, we have

$$\begin{aligned}\|\mathbf{w}^\star - \mathbf{v}\|_2^2 - \|\hat{\mathbf{w}} - \mathbf{v}\|_2^2 &= (w_i^\star - v_i)^2 + (w_j^\star - v_j)^2 - (w_i^\star + w_j^\star - v_i)^2 - (0 - v_j)^2 \\ &= 2w_j^\star(v_i - v_j - w_i^\star) \\ &> 2w_j^\star(v_i - v_j - z) \\ &\geq 0.\end{aligned}$$

$\hat{\mathbf{w}}$ has even smaller objective value than $\mathbf{w}^\star$, contradicting the optimality of $\mathbf{w}^\star$. Thus all $w_j^\star = 0$ for all $j \neq i$, which finishes the proof. ■

Proposition 2 *The dual solution $\lambda^\star$ of (2.3) satisfies*

$$0 \leq \lambda^\star \leq \frac{2\|\mathrm{vec}(G)\|_\infty + \|\mathbf{x}\|_\infty}{\min_{i \neq j}\{C_{ij}\}}. \tag{2.7}$$

Proof: By Danskin's theorem the dual problem (2.3) is differentiable in λ. Moreover, for any given $\tilde{\lambda}$, suppose the solution to the minimization is $\tilde{\Pi}$, then the gradient w.r.t. $\tilde{\lambda}$ is $\langle \tilde{\Pi}, C \rangle - \delta$.

Consider the i-th row of $G - \lambda C$. Assume on the contrary that

$$\lambda > \frac{2\|\mathrm{vec}(G)\|_\infty + \|\mathbf{x}\|_\infty}{\min_{i \neq j}\{C_{ij}\}}.$$

Then, for all $i \neq j$ we have (recall that $C_{ii} = 0$)

$$G_{ii} = G_{ii} - \lambda C_{ii} > G_{ij} - \lambda C_{ij} + x_i.$$

The condition in Lemma 1 is satisfied. A projection of $G - \lambda C$ results in a diagonal matrix $\tilde{\Pi}$. Thus

$$\begin{aligned} g'(\tilde{\lambda}) &= \langle \tilde{\Pi}, C \rangle - \delta \\ &= \sum_{i=j} \tilde{\Pi}_{ij} C_{ij} + \sum_{i \neq j} \tilde{\Pi}_{ij} C_{ij} - \delta \\ &= -\delta. \end{aligned}$$

The derivative is strictly negative, hence $\tilde{\lambda}$ is suboptimal, which finishes the proof.

■

Proposition 3 *The primal solution* $\Pi^\star$ *and the dual solution* $\lambda^\star$ *satisfies*

$$\Pi^\star = \underset{\Pi \mathbf{1} = \mathbf{x}, \Pi \geq 0}{\operatorname{argmin}} \ \|\Pi - G + \lambda^\star C\|_{\mathrm{F}}^2,$$

thus $\Pi^\star$ *can be computed in* $O(n^2 \log n)$ *time given* $\lambda^\star$.

Proof: This is a direct implication of KKT conditions. ■

B.2 Dual Linear Minimization Oracle without Entropic Regularization

Proposition 4 *The dual problem of* (2.9) *is*

$$\underset{\lambda \geq 0}{\text{maximize}} \ \ -\lambda\delta + \sum_{i=1}^{n} x_i \min_{1 \leq j \leq n} \Big(H_{ij} + \lambda C_{ij} \Big). \tag{2.10}$$

Proof: Introducing the Lagrange multiplier $\lambda \geq 0$ for the constraint $\langle \Pi, C \rangle \leq \delta$, we arrive at the following dual problem

$$\underset{\lambda \geq 0}{\text{maximize}} \ g(\lambda),$$

where

$$\begin{aligned} g(\lambda) &= \min_{\Pi \geq 0, \Pi \mathbf{1} = \mathbf{x}} \ \langle \Pi, H \rangle + \lambda \left(\langle \Pi, C \rangle - \delta \right) \\ &= \min_{\Pi \geq 0, \Pi \mathbf{1} = \mathbf{x}} \ \langle \Pi, H + \lambda C \rangle - \lambda \delta \\ &= -\lambda \delta + \sum_{i=1}^{n} x_i \min_{1 \leq j \leq n} \Big(H_{ij} + \lambda C_{ij} \Big). \end{aligned}$$

The last equality uses the fact that the constraints are independent for each row, thus the minimization is separable. ■

Proposition 5 *The dual solution $\lambda^\star$ of (2.10) satisfies*

$$0 \leq \lambda^\star \leq \frac{2 \left\| \mathrm{vec}(H) \right\|_\infty}{\min_{i \neq j} \{C_{ij}\}}. \tag{2.11}$$

Proof: A key observation is that the dual objective is a piece-wise linear function w.r.t. λ. We can roughly estimate the range of the maximizer, by analyzing the slope of this function.

Suppose

$$\lambda > \frac{2 \left\| \mathrm{vec}(H) \right\|_\infty}{\min_{i \neq j} \{C_{ij}\}}. \tag{B.1}$$

Then for all $i \neq j$, we have

$$\lambda C_{ij} > H_{ii} - H_{ij},$$

which implies $H_{ii} + \lambda C_{ii} < H_{ij} + \lambda C_{ij}$ for all $i \neq j$ (recall that $C_{ii} = 0$). Thus

$$\begin{aligned} g(\lambda) &= -\lambda \delta + \sum_{i=1}^{n} x_i \min_{1 \leq j \leq n} \Big(H_{ij} + \lambda C_{ij} \Big) \\ &= -\lambda \delta + \sum_{i=1}^{n} x_i \left(H_{ii} + \lambda C_{ii} \right) \\ &= -\lambda \delta + \sum_{i=1}^{n} x_i H_{ii} \end{aligned}$$

is a linear function with negative slope. Thus, any λ satisfies (B.1) cannot be a dual solution, which completes the proof. ■

B.3 Dual Linear Minimization Oracle with Dual Entropic Regularization

Proposition 6 *The dual problem of* (2.13) *is*

$$\underset{\lambda\geq 0}{\text{maximize}} -\lambda\delta+\gamma\sum_{i:x_i>0}x_i\log x_i-\gamma\sum_{i=1}^{n}x_i\log\sum_{j=1}^{n}\exp\left(-\frac{H_{ij}+\lambda C_{ij}}{\gamma}\right). \tag{2.14}$$

Proof: Introducing the Lagrange multiplier $\lambda\geq 0$ for the constraint $\langle\Pi,C\rangle\leq\delta$, we arrive at the following dual problem

$$\underset{\lambda\geq 0}{\text{maximize}}\ g(\lambda),$$

where

$$g(\lambda)=\min_{\Pi\geq 0,\Pi\mathbf{1}=\mathbf{x}}\langle\Pi,H\rangle+\gamma\sum_{i=1}^{n}\sum_{j=1}^{n}\Pi_{ij}\log\Pi_{ij}+\lambda(\langle\Pi,C\rangle-\delta) \tag{B.2}$$

$$=\min_{\Pi\geq 0,\Pi\mathbf{1}=\mathbf{x}}-\lambda\delta+\langle\Pi,H+\lambda C\rangle+\gamma\sum_{i=1}^{n}\sum_{j=1}^{n}\Pi_{ij}\log\Pi_{ij}. \tag{B.3}$$

Without loss of generality, we assume that x_i is strictly positive for all i. Otherwise, $x_i=0$ implies $\Pi_{ij}=0$ for all j, thus the i-th row of Π does not even appear in the minimization.

Notice that the inner minimization in (B.3) is separable, since the constraint on Π is independent for each row. For each row, the minimization is equivalent to a Kullback–Leibler projection to a simplex, which admits a closed form. For the sake of completeness, we give a derivation here. For the i-th row,

$$\begin{aligned}\sum_{j=1}^{n}\Pi_{ij}\left(H_{ij}+\lambda C_{ij}\right)+\gamma\sum_{j=1}^{n}\Pi_{ij}\log\Pi_{ij}&=\gamma\sum_{j=1}^{n}\Pi_{ij}\log\frac{\Pi_{ij}}{\exp\left(-\frac{H_{ij}+\lambda C_{ij}}{\gamma}\right)}\\&=\gamma\sum_{j=1}^{n}\Pi_{ij}\left(\log\frac{\Pi_{ij}/x_i}{\exp\left(-\frac{H_{ij}+\lambda C_{ij}}{\gamma}\right)/a}+\log x_i-\log a\right)\\&=\gamma\sum_{j=1}^{n}\Pi_{ij}\log\frac{\Pi_{ij}/x_i}{\exp\left(-\frac{H_{ij}+\lambda C_{ij}}{\gamma}\right)/a}+\gamma x_i\left(\log x_i-\log a\right)\\&\geq\gamma x_i\left(\log x_i-\log a\right),\end{aligned}$$

where $a = \sum_{j=1}^{n} \exp\left(-\frac{H_{ij}+\lambda C_{ij}}{\gamma}\right)$ is a normalization constant. The last inequality holds if and only if

$$\Pi_{ij} = x_i \frac{\exp\left(-\frac{H_{ij}+\lambda C_{ij}}{\gamma}\right)}{\sum_{j=1}^{n} \exp\left(-\frac{H_{ij}+\lambda C_{ij}}{\gamma}\right)}.$$

Plugging in the above expression finishes the proof. ■

Proposition 7 *The primal solution* $\Pi^\star$ *and the dual solution* $\lambda^\star$ *satisfy*

$$\Pi^\star_{ij} = x_i \cdot \frac{\exp\left(-\frac{H_{ij}+\lambda^\star C_{ij}}{\gamma}\right)}{\sum_{j=1}^{n} \exp\left(-\frac{H_{ij}+\lambda^\star C_{ij}}{\gamma}\right)}. \tag{2.15}$$

Proof: This is a direct implication of KKT conditions. See the KL projection derivation in the proof of Proposition 6 for a detailed explanation. ■

Proposition 8 *The dual solution* $\lambda^\star$ *of* (2.14) *satisfies*

$$0 \le \lambda^\star \le \frac{\left[2\left\|\mathrm{vec}(H)\right\|_\infty + \gamma \log \frac{1}{\delta}\mathbf{x}^\top C \mathbf{1}\right]_+}{\min_{i\neq j}\{C_{ij}\}}. \tag{2.16}$$

Proof: To begin with, we have the following bound on the derivative:

$$\begin{aligned}
g'(\lambda) &= -\delta + \sum_{i=1}^{n} x_i \frac{\sum_{j=1}^{n} \exp\left(-\frac{H_{ij}+\lambda C_{ij}}{\gamma}\right) C_{ij}}{\sum_{j=1}^{n} \exp\left(-\frac{H_{ij}+\lambda C_{ij}}{\gamma}\right)} \\
&= -\delta + \sum_{i=1}^{n} x_i \frac{\sum_{j=1}^{n} \exp\left(-\frac{H_{ij}+\lambda C_{ij}-H_{ii}}{\gamma}\right) C_{ij}}{\sum_{j=1}^{n} \exp\left(-\frac{H_{ij}+\lambda C_{ij}-H_{ii}}{\gamma}\right)} \\
&= -\delta + \sum_{i=1}^{n} x_i \frac{\sum_{j\neq i} \exp\left(-\frac{H_{ij}+\lambda C_{ij}-H_{ii}}{\gamma}\right) C_{ij}}{1 + \sum_{j\neq i} \exp\left(-\frac{H_{ij}+\lambda C_{ij}-H_{ii}}{\gamma}\right)} \\
&\le -\delta + \sum_{i=1}^{n} \sum_{j\neq i} x_i \exp\left(-\frac{H_{ij}+\lambda C_{ij}-H_{ii}}{\gamma}\right) C_{ij}.
\end{aligned}$$

The first equality uses translation invariance property of softmin function. The last inequality uses the fact that $C_{ii} = 0$. Notice that

$$\lambda > \frac{\left[2 \left\|\mathrm{vec}(H)\right\|_\infty + \gamma \log\left(\frac{1}{\delta}\mathbf{x}^\top C\mathbf{1}\right)\right]_+}{\min_{i\neq j}\{C_{ij}\}}$$

implies

$$\begin{aligned}\lambda C_{ij} &> 2\left\|\mathrm{vec}(H)\right\|_\infty + \gamma \log\left(\frac{1}{\delta}\mathbf{x}^\top C\mathbf{1}\right) \\ &\geq H_{ii} - H_{ij} + \gamma \log\left(\frac{1}{\delta}\mathbf{x}^\top C\mathbf{1}\right),\end{aligned}$$

for all $i \neq j$. Thus, we have

$$\exp\left(-\frac{H_{ij} + \lambda C_{ij} - H_{ii}}{\gamma}\right) < \frac{\delta}{\mathbf{x}^\top C\mathbf{1}}$$

for all $i \neq j$. Plug it back to $g'(\lambda)$. We have

$$\begin{aligned}g'(\lambda) &< -\delta + \frac{\delta}{\mathbf{x}^\top C\mathbf{1}} \cdot \sum_{i=1}^n \sum_{j=1}^n x_i C_{ij} \\ &= -\delta + \frac{\delta}{\mathbf{x}^\top C\mathbf{1}} \cdot \mathbf{x}^\top C\mathbf{1} \\ &= 0.\end{aligned}$$

The derivative is strictly negative hence λ cannot be optimal, which concludes the proof. ■

Appendix C

Further Experimental Details

In this section, we present further experimental details as well as some implementation details.

C.1 Models

Our MNIST and CIFAR-10, models are taken from (Wong et al. 2019). The MNIST model is a convolutional network with ReLU activations which achieves 98.89% clean accuracy. The CIFAR-10 model is a residual network with 94.76% clean accuracy. The ImageNet model is a ResNet-50 pretrained neural network, downloaded from PyTorch models subpackage, which achieves 72.0% top-1 clean accuracy on the first 100 samples from the validation set. All experiments are run on a single P100 GPU.

C.2 Stopping Criteria

Stopping criterion of projected Sinkhorn Denote $\text{obj}^{(t)}$ as the dual objective value of projected Sinkhorn in t-th iteration, we stop the algorithm upon the following condition is satisfied:

$$|\text{obj}^{(t+1)} - \text{obj}^{(t)}| \leq 10^{-4} + 10^{-4} \cdot \text{obj}^{(t)},$$

which is also used by Wong et al. (2019).

Stopping criterion of dual projection and dual LMO Both dual projection and dual LMO use the bisection method to solve dual problems. Bisection is terminated upon either

$$u - l \leq 10^{-4} \quad \text{or} \quad |g'(\tilde{\lambda})| \leq 10^{-4}$$

This condition lets us determine the 4-th digit after the decimal point of $\lambda^\star$, or the violation of transportation cost constraint is less than 10^{-4}. Note that a violation of 10^{-4} is extremely small, compared with $\delta = \epsilon \sum_{i=1}^n \mathbf{x}_i$, which is much (usually at least 10^5 times) larger than the tolerance since the pixel sum $\sum_{i=1}^n \mathbf{x}_i$ is usually a large number.

In practice, the upper bound (2.7) and (2.16) are often between 2 and 3 thanks to gradient normalization. Thus, the bisection method satisfies the stopping criterion in at most 15 iterations ($2 \times 2^{-15} \approx 10^{-4}$).

C.3 Step Sizes of PGD

PGD with projected Sinkhorn On MNIST, CIFAR-10 and ImageNet, the step sizes are set to 0.1. Notice that 0.1 is also the step size used by Wong et al. (2019) on MNIST and CIFAR-10. The gradient is normalized using ℓ_∞ norm:

$$\underset{\|v\|_\infty \leq 1}{\operatorname{argmax}}\, v^\top \nabla_{\mathbf{x}} \ell(\mathbf{x}, y) = \operatorname{sign}\left(\nabla_{\mathbf{x}} \ell(\mathbf{x}, y)\right).$$

Again, this is the same setting used by Wong et al. (2019). While $\eta = 1.0$ achieves lower adversarial accuracy on the first batch of samples in Appendix C.4, we find this large step size causes numerical overflow easily on the remaining batches. Thus we choose $\eta = 0.1$ to present the experimental results.

PGD with dual projection On MNIST, the step size is set to 0.1. On CIFAR-10 and ImageNet, the step size is set to 0.01. The gradient is normalized in the following way:

$$\frac{\nabla_\Pi \ell(\Pi^\top \mathbf{1}, y)}{\max_{i,j} |\nabla_\Pi \ell(\Pi^\top \mathbf{1}, y)|}.$$

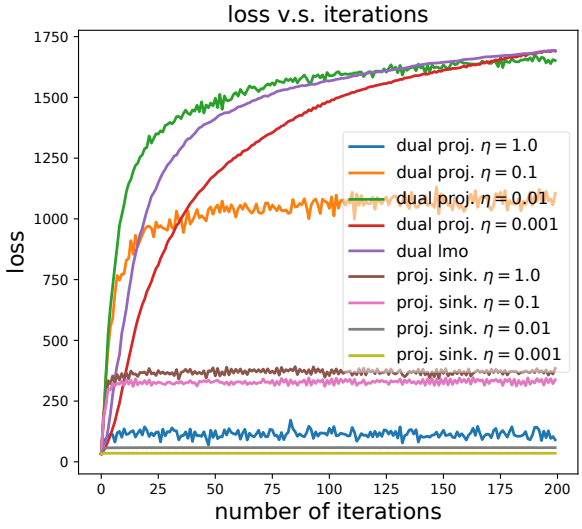
C.4 Further Results on Convergence of Outer Maximization

We plot the loss and adversarial accuracy w.r.t. the number of iterations in Figure C.1 ($\epsilon = 0.005$ on both CIFAR-10 and ImageNet). Dual LMO uses $\gamma = 10^{-3}$. Projected Sinkhorn uses $\gamma = 5 \cdot 10^{-5}$ on CIFAR-10 and $\gamma = 5 \cdot 10^{-6}$ on ImageNet.

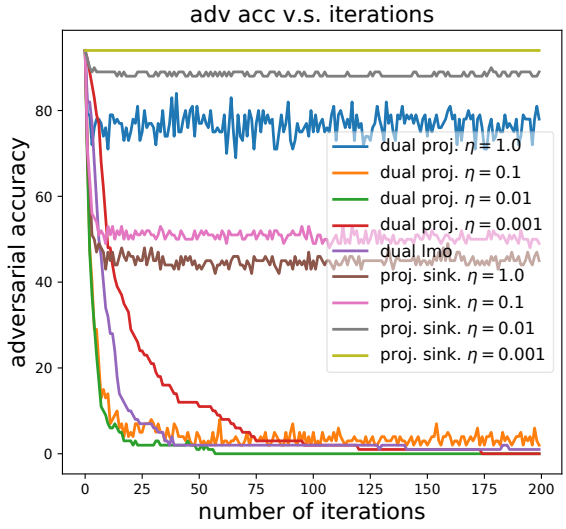
Frank-Wolfe with dual LMO FW uses the default decay schedule $\frac{2}{t+1}$. We observe that FW with dual LMO converges very fast especially at the initial stage, even when using the simple default decay schedule.

PGD with Projected Sinkhorn We observe that when η is small (*e.g.* $\eta = 0.01, 0.001$), PGD with projected Sinkhorn barely makes progress in the optimization. While aggressively large step sizes (*e.g.* 1.0 and 0.1) can make progress, the curves are very noisy indicating the steps sizes are too large, and the loss is still much lower than the other two attacks.

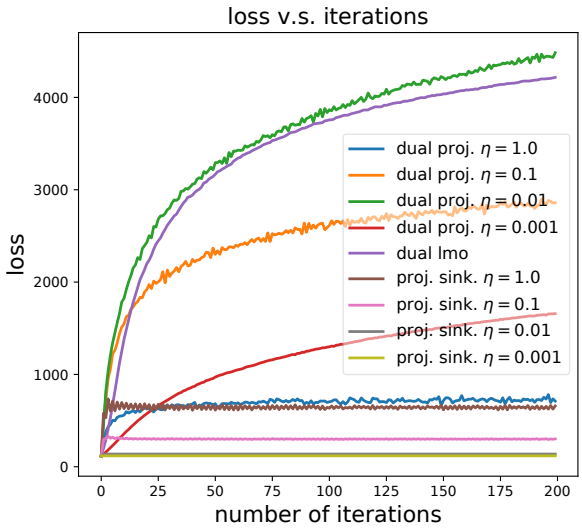
PGD with Dual Projection In contrast, PGD with dual projection has more meaningful curves: small η (*e.g.* 0.001) converges very slowly; large η (*e.g.* 1.0 and 0.1) makes the curves noisy, while appropriate choice of η (*e.g.* 0.01) always achieves the highest loss and also the lowest adversarial accuracy. In these cases, PGD with dual projection converges comparably and sometimes slightly faster than Frank-Wolfe.



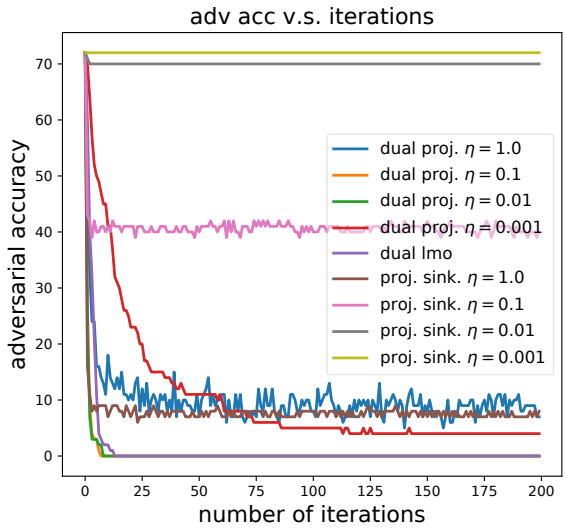
(a) loss w.r.t. iterations on CIFAR-10



(b) adversarial accuracy on CIFAR-10



(c) loss w.r.t. iterations on ImageNet



(d) adversarial accuracy on ImageNet

Figure C.1: Convergence of outer maximization of different attacks.

C.5 MNIST and CIFAR-10 Adversarial Examples

Wasserstein adversarial perturbations generated by PGD with dual projection and FW with dual LMO ($\gamma = 10^{-3}$) are very sparse. Hence, they do not reflect the shapes in the clean images. Perturbations reflect the shapes only when the entropic regularization introduces large approximation error (*e.g.* FW with dual LMO ($\gamma = 10$) and PGD with projected Sinkhorn).

For the sake of visualization of the approximation error, we let PGD use smaller step sizes ($\eta = 0.01$) and more iterations (1000) when combined with projected Sinkhorn.[1] We observe that using large step size, *e.g.*, $\eta = 0.1$, often generates blurry perturbations (not necessarily reflecting shapes clearly). But they are still much more dense compared with adversarial perturbations generated dual projection and dual LMO ($\gamma = 10^{-3}$).

[1] This is the same case for Figure 3.4 in the main paper. Notice that our normalization method is slightly different from that of Wong et al. (2019), which might account for the slight difference of the visualization results.

(a) MNIST (b) CIFAR-10

Figure C.2: Comparison of Wasserstein adversarial examples and Wasserstein perturbations generated by different attacks on MNIST ($\epsilon = 0.2$) and CIFAR-10 ($\epsilon = 0.002$). Predicted labels are shown on the top of images. Perturbations are scaled linearly to $[0, 1]$ for visualization.

C.6 ImageNet Adversarial Examples

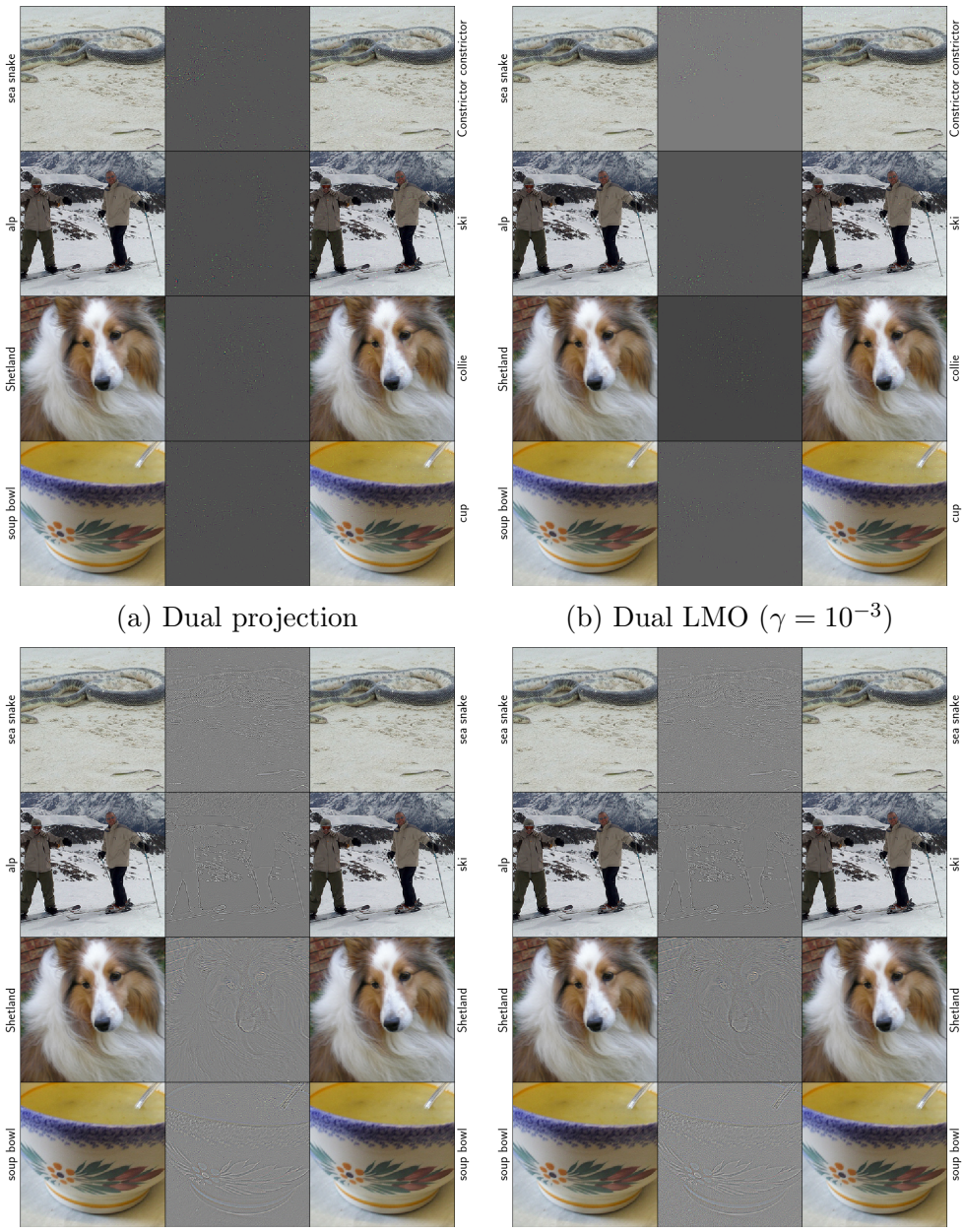
We visualize ImageNet adversarial examples in Figure C.3.

C.7 Analysis of Running Time of Projected Sinkhorn

One possible explanation for slow per iteration running speed of projected Sinkhorn is that, each iteration of projected Sinkhorn requires solving n nonlinear univariate equations. In implementation, this step is done by calling Lambert function, which eventually calls Halley's method, a root-finding algorithm. While solving each nonlinear equation is counted as $O(1)$ in analysis, this operation is potentially much more expensive than other basic arithmetic operations (*e.g.* addition and multiplication). In contrast, both dual projection and dual linear minimization only rely on more standard arithmetic operations (*e.g.* multiplication, division and comparison, logarithm and exponential functions).

We test the running time of the nonlinear equation solvers in experiments. On a single RTX 2080 Ti GPU, for a batch of 100 samples on MNIST, we observe that Lambert function evaluation takes about 26% of the running time during each iteration of projected Sinkhorn. For a batch of 100 samples on CIFAR-10, Lambert function evaluation takes about 28% of the running time during each iteration of projected Sinkhorn. For a batch of 50 samples on ImageNet, Lambert function evaluation takes about 32% of the running time of one iteration of projected Sinkhorn.

We note that GPU time recording is somewhat inconsistent on different machines. In our case, all experiments in the main paper are run on a single P100 GPU on a cluster. However, on a 2080 Ti local GPU, we observe that projected Sinkhorn is slightly faster than dual projection on ImageNet in terms of per iteration running time (but still at least twice slower than dual LMO). While on MNIST and CIFAR-10, dual projection and dual LMO are consistently faster than projected Sinkhorn, both on P100 cluster and 2080 Ti local machine.



(a) Dual projection

(b) Dual LMO ($\gamma = 10^{-3}$)

(c) Dual LMO ($\gamma = 10$)

(d) Projected Sinkhorn ($\gamma = 10^{-4}$)

Figure C.3: Comparison of Wasserstein adversarial examples generated by different attacks ($\epsilon = 0.005$). Notice that adversarial pertubations reflect shapes in the images only when the entropic regularization is large (bottom two subfigures).

C.8 Case Study: Feasibility of Adversarial Examples

In this section, we present a sanity check of feasibility of adversarial examples generated by different algorithms on MNIST.

Table C.1: A sanity check of feasibility of Wasserstein adversarial examples generated by different algorithms on MNIST. **2nd column:** The average Wasserstein distance between adversarial examples and clean images (the higher the better). **3rd column:** The maximum pixel value in the generated adversarial examples (the lower the better). **4th column:** The percentage of pixel mass that exceeds 1 (the lower the better). The third and the forth columns are largest values over a mini-batch, while the second column is the average over a mini-batch.

method	$\mathcal{W}(\mathbf{x}, \mathbf{x}_{adv})$	$\max_i \{\mathbf{x}_i\}$	$\frac{\sum_{i=1}^n \max\{\mathbf{x}_i - 1, 0\}}{\sum_{i=1}^n \mathbf{x}_i}$
PGD + Proj. Sink. ($\gamma = 1/1000$)	0.109965	1.474048	3.646899%
PGD + Dual Proj. (w/o post-processing)	0.493146	4.118621	15.812986%
PGD + Dual Proj.	0.444885	1.000030	0.000034%
FW + Dual LMO (w/o post-processing)	0.428238	3.955514	10.406417%
FW + Dual LMO	0.399014	1.000015	0.000025%

Setup We test all attacks on the first 100 samples on the test set of MNIST with $\epsilon = 0.5$. All parameter settings are the same as the table in the main paper. Dual LMO uses $\gamma = 10^{-3}$ and projected Sinkhorn uses $\gamma = 1/1000$.

Wasserstein Constraint The (exact) Wasserstein distances presented in Table C.1 are calculated by a linear programming solver. We observe that all adversarial examples strictly satisfy the the Wasserstein constraint $\mathcal{W}(\mathbf{x}, \mathbf{x}_{adv}) \leq 0.5$. We also observe that the Wasserstein adversarial examples generated by PGD with dual projection and FW with dual LMO have much larger Wasserstein distance than those of projected Sinkhorn. This again hightlights that projected Sinkhorn is only an approximate projection operator, thus PGD cannot fully explore the Wasserstein ball, resulting in a weak attack.

Hypercube Constraint Without post-processing, all attacks generate Wasserstein adversarial examples that violate the hypercube constraint a lot. However, after applying our post-processing algorithm, the generated Wasserstein adversarial examples roughly satisfy the hypercube constraint $[0, 1]^n$ up to a reasonable precision.