A Riemannian Block Coordinate Descent Method for Computing the Projection Robust Wasserstein Distance

Minhui Huang [1] Shiqian Ma [2] Lifeng Lai [1]

Abstract

The Wasserstein distance has become increasingly important in machine learning and deep learning. Despite its popularity, the Wasserstein distance is hard to approximate because of the curse of dimensionality. A recently proposed approach to alleviate the curse of dimensionality is to project the sampled data from the high dimensional probability distribution onto a lower-dimensional subspace, and then compute the Wasserstein distance between the projected data. However, this approach requires to solve a max-min problem over the Stiefel manifold, which is very challenging in practice. In this paper, we propose a Riemannian block coordinate descent (RBCD) method to solve this problem, which is based on a novel reformulation of the regularized max-min problem over the Stiefel manifold. We show that the complexity of arithmetic operations for RBCD to obtain an ϵ-stationary point is $O(\epsilon^{-3})$, which is significantly better than the complexity of existing methods. Numerical results on both synthetic and real datasets demonstrate that our method is more efficient than existing methods, especially when the number of sampled data is very large.

1. Introduction

The Wasserstein distance measures the closeness of two probability distributions on a given metric space. It has a broad range of applications in machine learning problems, including the latent mixture models (Ho & Nguyen, 2016), representation learning (Ozair et al., 2019), reinforcement learning (Bellemare et al., 2017) and stochastic optimization (Nagaraj et al., 2019). Intuitively, the Wasserstein distance is the minimum cost of turning one distribution into the other. To calculate the Wasserstein distance, one is required to solve an optimal transport (OT) problem, which has been widely adopted in machine learning and data science.

However, it is known that the sample complexity of approximating Wasserstein distances using only samples can grow exponentially in dimension (Dudley, 1969; Fournier & Guillin, 2015; Weed & Bach, 2019; Lei, 2020). This leads to very large-scale OT problems that are challenging to solve using traditional approaches. As a result, this has motivated research on mitigating this curse of dimensionality when approximating Wasserstein distance using OT. One approach for reducing the dimensionality is the sliced approximation of OT proposed by (Rabin et al., 2011). This approach projects the clouds of points from two probability distributions onto a given line, and then computes the OT cost between these projected values as an approximation to the original OT cost. This idea has been further studied in (Kolouri et al., 2016; Bonneel et al., 2015; Kolouri et al., 2019; Deshpande et al., 2019; Nguyen et al., 2021a;b) for defining kernels, computing barycenters, and training generative models. Recently, motivated by the sliced approximation of OT, (Paty & Cuturi, 2019) and (Niles-Weed & Rigollet, 2019) proposed to project the distance measures onto k-dimensional subspaces. The k-dimensional subpsaces are obtained by maximizing the Wasserstein distance between two measures after projection. The approach is called Wasserstein projection pursuit (WPP), and the largest Wasserstein distance between the two measures after projection onto the k-dimensional subspaces is called the projection robust Wasserstein distance (PRW). As proved in (Niles-Weed & Rigollet, 2019) and (Lin et al., 2020b), WPP/PRW indeed reduces the sample complexity and resolves the issue of curse of dimensionality for the spiked transport model. However, computing PRW requires solving a nonconvex max-min problem over the Stiefel manifold, which demands efficient algorithms. In this paper, we propose a novel algorithm that can compute PRW efficiently and faithfully.

In the case of discrete probability measures, one is given two sets of finite number atoms, $\{x_1, x_2, \ldots, x_n\} \subset \mathbb{R}^d$ and $\{y_1, y_2, \ldots, y_n\} \subset \mathbb{R}^d$, and two probability distributions $\mu_n = \sum_{i=1}^n r_i \delta_{x_i}$ and $\nu_n = \sum_{j=1}^n c_j \delta_{y_j}$. Here $r = (r_1, r_2, \ldots, r_n)^\top \in \Delta^n$ and $c = (c_1, c_2, \ldots, c_n)^\top \in \Delta^n$,

[1] Department of Electrical and Computer Engineering, University of California, Davis, CA, USA [2] Department of Mathematics, University of California, Davis, CA, USA. Correspondence to: Shiqian Ma <sqma@ucdavis.edu>.

Proceedings of the 38th International Conference on Machine Learning, PMLR 139, 2021. Copyright 2021 by the author(s).

Δ^n denotes the probability simplex in $\mathbb{R}^n$ and δ_x denotes the Dirac delta function at x. Computing the Wasserstern distance between μ_n and ν_n is equivalent to solving an OT problem (Villani, 2008):

$$\mathcal{W}^2(\mu_n,\nu_n) \;=\; \min_{\pi\in\Pi(\mu_n,\nu_n)} \langle C,\pi\rangle, \tag{1}$$

where the transporation polytope $\Pi(\mu_n,\nu_n) := \{\pi \in \mathbb{R}_+^{n\times n} \mid \pi\mathbf{1} = r, \pi^\top\mathbf{1} = c\}$, and $\mathbf{1}$ denotes the n-dimensional all-one vector. Throughout this paper, $C \in \mathbb{R}^{n\times n}$ denotes the matrix whose (i,j)-th component is $C_{ij} = \|x_i - y_j\|^2$. Computing the PRW can be cast as the following max-min problem (Paty & Cuturi, 2019):

$$\begin{aligned}\mathcal{P}_k^2(\mu_n,\nu_n) := \max_{U\in\mathcal{M}} \min_{\pi\in\Pi(\mu_n,\nu_n)} f(\pi,U) :=&\\ \sum_{i,j=1}^{n} \pi_{ij}\|U^\top x_i - U^\top y_j\|^2.&\end{aligned} \tag{2}$$

Throughout this paper, $\mathcal{M}$ denotes the Stiefel manifold $\mathcal{M} \equiv \mathrm{St}(d,k) := \{U \in \mathbb{R}^{d\times k} \mid U^\top U = I_{k\times k}\}$. Here integer $k \in [d]$, where $[d]$ denotes the set of integers $\{1,2,\ldots,d\}$. Note that $\|U^\top x_i - U^\top y_j\|^2$ is the distance between the projected x_i and y_j and U denotes a basis of the k-dimensional subspace. Due to its nonconvex nature, solving (2) is not an easy task. In fact, (Paty & Cuturi, 2019) concluded that the PRW (2) is difficult to compute, and they proposed to solve its corresponding dual problem – the subspace robust Wasserstein distance (SRW):

$$\mathcal{S}_k^2(\mu_n,\nu_n) := \min_{\pi\in\Pi(\mu_n,\nu_n)} \max_{U\in\mathcal{M}} f(\pi,U), \tag{3}$$

where f is defined in (2). It is shown in (Paty & Cuturi, 2019) that the SRW (3) is equivalent to:

$$\begin{aligned}\mathcal{S}_k^2(\mu_n,\nu_n) = \max_{0\preceq\Omega\preceq I, \mathrm{Tr}(\Omega)=k} s(\Omega) :=&\\ \min_{\pi\in\Pi(\mu_n,\nu_n)} \sum_{ij} \pi_{ij}(x_i-y_j)^\top\Omega(x_i-y_j),&\end{aligned} \tag{4}$$

which can be viewed as maximizing the concave function $s(\Omega)$ over the convex set $\{\Omega \mid 0 \preceq \Omega \preceq I, \mathrm{Tr}(\Omega) = k\}$, where $\mathrm{Tr}(\Omega)$ denotes the trace of matrix Ω. Problem (4) is a convex optimization problem and thus numerically more tractable. (Paty & Cuturi, 2019) proposed a projected subgradient method for solving (3), and in each iteration computing the subgradient of s requires solving an OT problem in the form of (1). To improve the computational efficiency, they also proposed a Frank-Wolfe method for solving the following entropy-regularized SRW:

$$\begin{aligned}&\max_{0\preceq\Omega\preceq I, \mathrm{Tr}(\Omega)=k} s_\eta(\Omega) :=\\ &\min_{\pi\in\Pi(\mu_n,\nu_n)} \sum_{ij} \pi_{ij}(x_i-y_j)^\top\Omega(x_i-y_j) - \eta H(\pi),\end{aligned} \tag{5}$$

where $H(\pi) = -\sum_{ij}(\pi_{ij}\log\pi_{ij} - \pi_{ij})$ is the entropy regularizer and $\eta > 0$ is a weighting parameter. Each iteration of the Frank-Wolfe method requires solving a regularized OT (REGOT) problem in the following form:

$$\min_{\pi\in\Pi(\mu_n,\nu_n)} \langle M,\pi\rangle - \eta H(\pi), \tag{6}$$

for a given matrix $M \in \mathbb{R}^{n\times n}$. Solving (6) can be done more efficiently using the Sinkhorn's algorithm (Cuturi, 2013). However, note that solving (3) does not yield a solution to (2) because there exists a duality gap.

More recently, (Lin et al., 2020a) proposed a Riemannian gradient method to compute the PRW (2). More specifically, they proposed the RGAS (Riemannian Gradient Ascent with Sinkhorn Iteration) algorithm for computing the PRW with entropy regularization:

$$\begin{aligned}\max_{U\in\mathcal{M}} p(U) := \min_{\pi\in\Pi(\mu_n,\nu_n)} f_\eta(\pi,U) :=&\\ \sum_{ij}\pi_{ij}\|U^\top x_i - U^\top y_j\|^2 - \eta H(\pi).&\end{aligned} \tag{7}$$

(Lin et al., 2020a) proved that the RGAS algorithm combined with a rounding procedure (will be discussed later) gives an ϵ-stationary point to the PRW problem (2). A typical iteration of RGAS can be described as:

$$\begin{aligned}\pi^{t+1} &:= \mathrm{REGOT}(\{(x_i,r_i)\}_{i\in[n]}, \{(y_j,c_j)\}_{j\in[n]}, U^t, \eta)\\ \xi^{t+1} &:= \mathrm{grad}\, p(U^t)\\ U^{t+1} &:= \mathrm{Retr}_{U^t}(\tau\xi^{t+1}).\end{aligned}$$

Here $\mathrm{grad}\, p$ denotes the Riemannian gradient of function p, Retr denotes a retraction operator on the manifold $\mathcal{M}$, and π^{t+1} is the optimal solution to the REGOT problem (6) with $M_{ij} = \|(U^t)^\top(x_i - y_j)\|_2^2$. Note that computing ξ^{t+1} in fact requires π^{t+1}, and the latter further requires to solve a REGOT problem (6). This can be costly because an iterative solver for REGOT is needed in every iteration.

Our contributions. Motivated by the demand for efficient algorithms for computing PRW (2), we design a novel Riemannian block coordinate descent (RBCD) algorithm for solving this problem, and analyze its convergence behavior. Our main contributions of this paper lie in several folds. (i) We propose an equivalent formulation of (7), which consists a minimization problem only and thus is much easier to solve than the max-min problem (7). (ii) We propose a RBCD algorithm for solving the equivalent formulation of (7). The per-iteration complexity of RBCD is much lower than the existing methods in (Paty & Cuturi, 2019) and (Lin et al., 2020a), as it does not need to solve OT or REGOT problems. This makes our algorithm suitable for large-scale problems. (iii) We propose a variant of RBCD (named RABCD) in the supplementary material that adopts an adaptive step size for the Riemannian gradient step. This stategy helps speed up the convergence of RBCD in practice. (iv) We prove that the complexity of arithmetic operations of RBCD and RABCD are both $O(\epsilon^{-3})$ for obtaining an ϵ-stationary point of problem (2). This significantly improves the corresponding complexity of RGAS, which is $O(\epsilon^{-12})$.

Notation. We denote the tangent space of a Riemannian manifold $\mathcal{M}$ as $\mathrm{T}_U\mathcal{M}$. The Riemannian metric induced from the Euclidean inner product is $\langle \xi, \eta \rangle_U = \mathrm{Tr}(\xi^\top \eta), \forall \xi, \eta \in \mathrm{T}_U\mathcal{M}$.

2. A Riemannian Block Coordinate Descent Algorithm for Computing the PRW

In this section, we present our RBCD algorithm for computing the PRW (2). Our algorithm is based on a new reformulation of the entropy-regularized problem (7). First, we introduce some notation for the ease of the presentation. We denote $\varphi(\pi) := \pi\mathbf{1}$, and $\kappa(\pi) := \pi^\top\mathbf{1}$. The inner minimization problem in (7) can be equivalently written as

$$\begin{aligned} &\min_{\pi} \sum_{ij} \pi_{ij} \|U^\top x_i - U^\top y_j\|^2 - \eta H(\pi), \\ &\text{s.t.}, \varphi(\pi) = r, \kappa(\pi) = c, \end{aligned} \tag{8}$$

which is a convex problem with respect to π. The Lagrangian dual problem of (8) is given by:

$$\begin{aligned} \max_{\alpha,\beta} \min_{\pi} &\sum_{ij} \pi_{ij} \|U^\top x_i - U^\top y_j\|^2 - \eta H(\pi) \\ &+ \alpha^\top(\varphi(\pi) - r) + \beta^\top(\kappa(\pi) - c), \end{aligned} \tag{9}$$

where α and β denote the Lagrange multipliers of the two equality constraints. It is easy to verify that the optimality conditions of the minimization problem in (9) are given by:

$$0 = \|U^\top x_i - U^\top y_j\|^2 + \eta \log \pi_{ij} + \alpha_i + \beta_j, \ \forall 1 \le i, j \le n,$$

from which we know that

$$\pi_{ij} = \exp((-\alpha_i - \beta_j - \|U^\top x_i - U^\top y_j\|^2)/\eta). \tag{10}$$

Substituting (10) into (9), we know that (9) is equivalent to

$$\begin{aligned} \max_{\alpha,\beta} \ -\eta \sum_{ij} \exp\left(-\frac{\alpha_i + \beta_j + \|U^\top x_i - U^\top y_j\|^2}{\eta}\right) \\ - \sum_i r_i \alpha_i - \sum_j c_j \beta_j. \end{aligned} \tag{11}$$

By combining (11) and (7), we know that (7) is equivalent to the following maximization problem:

$$\begin{aligned} \max_{U \in \mathcal{M}, \alpha, \beta} -\eta \sum_{ij} \exp\left(-\frac{\alpha_i + \beta_j + \|U^\top x_i - U^\top y_j\|^2}{\eta}\right) \\ - \sum_i r_i \alpha_i - \sum_j c_j \beta_j. \end{aligned} \tag{12}$$

We now define $u = -\alpha/\eta, v = -\beta/\eta$, and function $\pi(u, v, U) \in \mathbb{R}^{n \times n}$ with

$$[\pi(u, v, U)]_{ij} := \exp\left(-\frac{1}{\eta}\|U^\top(x_i - y_j)\|^2 + u_i + v_j\right). \tag{13}$$

Then (12) is equivalent to the following problem:

$$\min_{U \in \mathcal{M}, u, v \in \mathbb{R}^n} g(u, v, U) := \sum_{i,j} [\pi(u, v, U)]_{ij} - r^\top u - c^\top v. \tag{14}$$

There are three block variables (u, v, U) in (14), and the objective function g is a smooth function with respect to (u, v, U). Moreover, for fixed v and U, minimizing g with respect to u can be done analytically, and similarly, for fixed u and U, minimizing g with respect to v can also be done analytically. For fixed u and v, minimizing g with respect to U is a Riemannian optimization problem with smooth objective function. Therefore, we propose a Riemannian block coordinate descent method for solving (14), whose t-th iteration updates the iterates as follows:

$$u^{t+1} := \min_u \ g(u, v^t, U^t) \tag{15a}$$

$$v^{t+1} := \min_v \ g(u^{t+1}, v, U^t) \tag{15b}$$

$$\begin{aligned} V_{\pi(u^{t+1}, v^{t+1}, U^t)} := &\sum_{ij} [\pi(u^{t+1}, v^{t+1}, U^t)]_{ij} \\ &\cdot (x_i - y_j)(x_i - y_j)^\top \end{aligned} \tag{15c}$$

$$\begin{aligned} \xi^{t+1} := &\mathrm{grad}_U g(u^{t+1}, v^{t+1}, U^t) = \\ &\mathrm{Proj}_{\mathrm{T}_{U^t}\mathcal{M}}\left(-\frac{2}{\eta} V_{\pi(u^{t+1}, v^{t+1}, U^t)} U^t\right) \end{aligned} \tag{15d}$$

$$U^{t+1} := \mathrm{Retr}_{U^t}(-\tau \xi^{t+1}), \tag{15e}$$

where the notation V_π is defined as: $V_\pi = \sum_{ij} \pi_{ij}(x_i - y_j)(x_i - y_j)^\top \in \mathbb{R}^{d \times d}$. Note that (15a) and (15b) admit closed-form solutions given by

$$u^{t+1} = u^t + \log(r./\varphi(\pi(u^t, v^t, U^t))) \tag{16}$$

$$v^{t+1} = v^t + \log(c./\kappa(\pi(u^{t+1}, v^t, U^t))), \tag{17}$$

where $a./b$ denotes the component-wise division ofvectors a and b. Moreover, it is easy to verify:

$$\nabla_U g(u, v, U) = -\frac{2}{\eta} V_{\pi(u,v,U)} U. \tag{18}$$

Therefore, (15c)-(15e) give a Riemannian gradient step of g with respect to variable U. Also note that (15c) requires to compute $\pi(u^{t+1}, v^{t+1}, U^t)$, which can be computed using (13). The algorithm is terminated when the following stopping criterion is satisfied:

$$\begin{aligned} &\|\xi^{t+1}\|_F \le \frac{\epsilon_1}{4\eta}, \quad \|r - \varphi(\pi(u^t, v^t, U^t))\|_2 \le \frac{\epsilon_2}{8\|C\|_\infty}, \\ &\|c - \kappa(\pi(u^{t+1}, v^t, U^t))\|_1 \le \frac{\epsilon_2}{8\|C\|_\infty}, \end{aligned} \tag{19}$$

where ϵ_1, ϵ_2 are pre-given accuracy tolerances. The reason of using this stopping criterion will be clear in the convergence analysis later.

It should be pointed out that the optimal transportation plan π of (2) is not directly computed by RBCD, because the sequence $\pi(u^{t+1}, v^{t+1}, U^t)$ generated in (15) does not satisfy the constraints in (2). Therefore, a procedure is needed to compute an approximate solution π to the original problem (2). Here we adopt the rounding procedure proposed in (Altschuler et al., 2017), which is outlined in Algorithm 1, where the notation $a \wedge b$ picks the smaller value between a and b. Our final transportation plan is computed by rounding $\pi(u^{t+1}, v^t, U^t)$ using Algorithm 1. Combining this rounding procedure and the RBCD outlined above, we arrive at our final algorithm for solving the original PRW problem (2), which is detailed in Algorithm 2.

Algorithm 1 $Round(\pi, r, c)$

```
Input: $\pi \in \mathbb{R}^{n\times n}$, $r \in \mathbb{R}^n$, $c \in \mathbb{R}^n$.
$X = \mathrm{Diag}\,(x)$ with $x_i = \frac{r_i}{\varphi(\pi)_i} \wedge 1$
$\pi' = X\pi$
$Y = \mathrm{Diag}\,(y)$ with $y_j = \frac{c_i}{\kappa(\pi')_j} \wedge 1$
$\pi'' = \pi' Y$
$err_r = r - \varphi(\pi''), err_c = c - \kappa(\pi'')$
Output: $\pi'' + err_r err_c^\top / \|err_r\|_1$.
```

Algorithm 2 Riemannian Block Coordinate Descent Algorithm (RBCD)

```
Input: $\{(x_i, r_i)\}_{i\in[n]}$ and $\{(y_j, c_j)\}_{j\in[n]}$, $U^0 \in \mathcal{M}$, $u^0, v^0 \in \mathbb{R}^n$, and accuracy tolerance $\epsilon_1 \geq \epsilon_2 > 0$. Set parameters ($L_1$ and $L_2$ are defined in Proposition A.3) $\tau = \frac{1}{4L_2\|C\|_\infty/\eta + \rho L_1^2}$, $\eta = \frac{\epsilon_2}{4\log(n)+2}$, $\rho = \frac{2\|C\|_\infty}{\eta} + \frac{4\sqrt{k}\|C\|_\infty^2}{\eta^2}$.
for $t = 0, 1, 2, \ldots,$ do
    Compute $u^{t+1}$ by (16)
    Compute $v^{t+1}$ by (17)
    Compute $V_{\pi(u^{t+1}, v^{t+1}, U^t)}$, $\xi^{t+1}$ and $U^{t+1}$ by (15c)-(15e)
    if (19) is satisfied then
        break
    end if
end for
Output: $\hat{u} = u^{t+1}$, $\hat{v} = v^t$, $\hat{U} = U^t$, and $\hat{\pi} = Round(\pi(\hat{u}, \hat{v}, \hat{U}), r, c)$.
```

Remark 2.1 *We remark that* (16) *and* (17) *are the steps in the Sinkhorn's algorithm* (*Cuturi, 2013*). *It is easy to verify the following identities for any $t \geq 0$:*

$$\varphi(\pi(u^{t+1}, v^t, U^t)) = r, \ \kappa(\pi(u^{t+1}, v^{t+1}, U^t)) = c, \quad (20)$$

$$\|\pi(u^{t+1}, v^t, U^t)\|_1 = \|\pi(u^{t+1}, v^{t+1}, U^t)\|_1 = 1. \quad (21)$$

3. Convergence Analysis

In this section, we show that $(\hat{\pi}, \hat{U})$ returned by Algorithm 2 is an ϵ-stationary point of the PRW problem (2). We will also analyze its iteration complexity and complexity of arithmetic operations for obtaining such a point. We include all proofs in the supplementary material. The ϵ-stationary point for problem (2) is defined as follows.

Definition 3.1 *We call $(\hat{\pi}, \hat{U}) \in \Pi(\mu, \nu) \times \mathcal{M}$ an (ϵ_1, ϵ_2)-stationary point of the PRW problem* (2)*, if the following inequalities hold:*

$$\|\mathrm{grad}_U f(\hat{\pi}, \hat{U})\|_F \leq \epsilon_1, \quad (22a)$$

$$f(\hat{\pi}, \hat{U}) - \min_{\pi\in\Pi(\mu,\nu)} f(\pi, \hat{U}) \leq \epsilon_2. \quad (22b)$$

Remark 3.2 *(Lin et al., 2020a) defined the ϵ-stationary point of PRW* (2) *as the pair $(\hat{\pi}, \hat{U}) \in \Pi(\mu, \nu) \times \mathcal{M}$ that satisfies:*

$$\mathrm{dist}(0, \mathrm{subdiff}\, f(\hat{U})) \leq \epsilon \quad (23a)$$

$$f(\hat{\pi}, \hat{U}) - \min_{\pi\in\Pi(\mu,\nu)} f(\pi, \hat{U}) \leq \epsilon, \quad (23b)$$

where subdiff *denotes the Riemannian subgradient, and*

$$f(U) := \min_{\pi\in\Pi(\mu_n,\nu_n)} f(\pi, U) := \sum_{i,j=1}^{n} \pi_{ij}\|U^\top x_i - U^\top y_j\|^2.$$

In the appendix, we will show that our (22) *implies* (23) *up to a constant factor when $\epsilon_1 = \epsilon_2 = \epsilon$.*

We now present the complexity of RBCD.

Theorem 3.3 *The RBCD returns an (ϵ_1, ϵ_2)-stationary point of the PRW problem* (2) *in*

$$O\left((n^2dk + dk^2 + k^3)\log(n)\sqrt{k}\left(\frac{1}{\epsilon_2^3} + \frac{1}{\epsilon_1^2\epsilon_2}\right)\right) \quad (24)$$

arithmetic operations, where the $O(\cdot)$ hides constants related to L_1, L_2 and $\|C\|_\infty$ only.

Remark 3.4 *Note that the complexity of arithmetic operations of our RBCD is significantly better than the corresponding complexity of RGAS, which is*

$$O(n^2d\|C\|_\infty^4\epsilon^{-4} + n^2\|C\|_\infty^8\epsilon^{-8} + n^2\|C\|_\infty^{12}\epsilon^{-12}).$$

When $\epsilon_1 = \epsilon_2 = \epsilon$, our complexity bound (24) *reduces to*

$$O((n^2dk + dk^2 + k^3)\log(n)\sqrt{k}\epsilon^{-3}).$$

Since $k = O(1)$, we conclude that our complexity bound is significantly better than that of RGAS (Lin et al., 2020a).

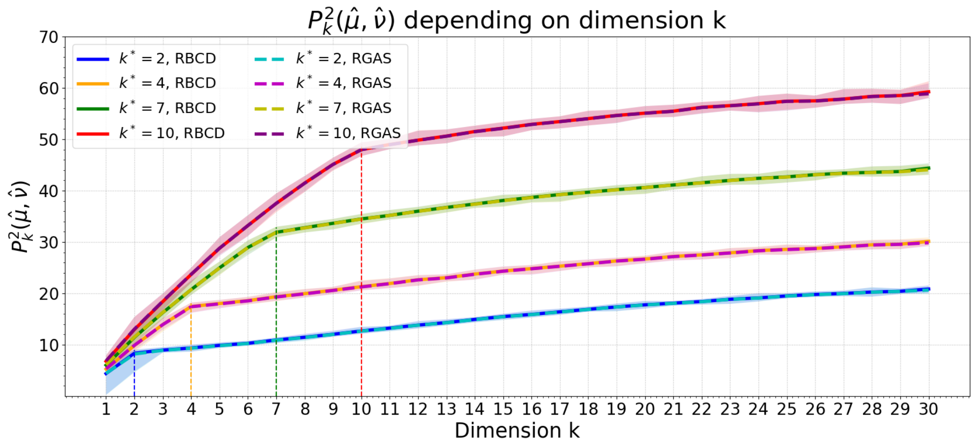

Figure 1. Computation of PRW value $\mathcal{P}_k^2(\hat{\mu}, \hat{\nu})$ depending on the dimension $k \in [d]$ and $k^* \in \{2, 4, 7, 10\}$, where $\hat{\mu}$ and $\hat{\nu}$ stand for the empirical measures of μ and ν with $n = 100, d = 30$. The solid and dash curves are the computation of $\mathcal{P}_k^2(\hat{\mu}, \hat{\nu})$ with the RBCD and RGAS algorithms, respectively. Each curve is the mean over 100 samples with shaded area covering the min and max values.

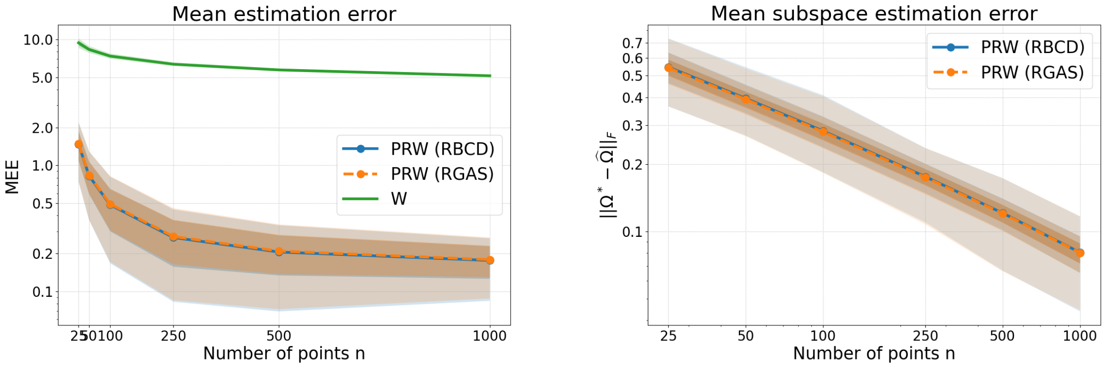

Figure 2. **Left:** The mean estimation error (MEE); **Right:** The mean subspace estimation error against the number of samples $n \in \{25, 50, 100, 250, 500, 1000\}$. We set $k^* = 2, d = 30$ and calculate the mean estimation error as $MEE = |\mathcal{P}_k^2(\hat{\mu}, \hat{\nu}) - 4k^*|$ for the PRW distance and $MEE = |\mathcal{W}_k^2(\hat{\mu}, \hat{\nu}) - 4k^*|$ for the Wasserstein distance. The subspace projection is calculated as $\hat{\Omega} = \hat{U}\hat{U}^\top$ in each run. The shaded areas represent the 10%-90% and 25%-75% quantiles over 500 samples. The shaded areas for two PRW algorithms are almost the same.

4. Numerical Experiments

In this section, we evaluate the performance of our proposed RBCD algorithm on calculating the PRW distance for both synthetic and real datasets[1]. We mainly focus on the comparison of the computational time between the RBCD algorithm and the RGAS algorithm (Lin et al., 2020a), which is currently the state of the art algorithm for solving the PRW problem. We use QR factorization as the retraction operation: $\mathrm{Retr}_U(\xi) = \mathrm{qf}(U + \xi)$ where $\mathrm{qf}(Z)$ denotes the Q-factor of the QR factorization of Z. All experiments in this section are implemented in Python 3.7 on a Linux server with an 32-core Intel Xeon CPU (E5-2667, v4, 3.20GHz per core).

4.1. Synthetic Dataset

We first focus on a synthetic example, which is adopted from (Paty & Cuturi, 2019; Lin et al., 2020a) and its ground truth Wasserstein distance can be computed analytically.

Fragmented Hypercube: We consider a uniform distribution over an hypercube $\mu = \mathcal{U}([-1, 1]^d)$ and a pushforward $\nu = T_{\#}\mu$ defined under the map $T(x) = x + 2\mathrm{sign}(x) \odot (\sum_{k=1}^{k^*} e_k)$, where $\mathrm{sign}(\cdot)$ is taken elementwise, $k^* \in [d]$ and e_i, $i \in [d]$ is the canonical basis of $\mathbb{R}^d$. The pushforward T splits the hypercube into 2^{k^*} different hyper rectangles. Since T can be viewed as the subgradient of a convex function, (Brenier, 1991) has shown that T is an optimal transport map between μ and $\nu = T_{\#}\mu$ with $\mathcal{W}(\mu, \nu)^2 = 4k^*$. In this case, the displacement vector $T(x) - x$ lies in the k^*-dimensional subspace spanned by $\{e_j\}_{j \in [k^*]}$ and we should have $\mathcal{P}_k^2 = 4k^*$ for any

[1] Code available at https://github.com/mhhuang95/PRW_RBCD.

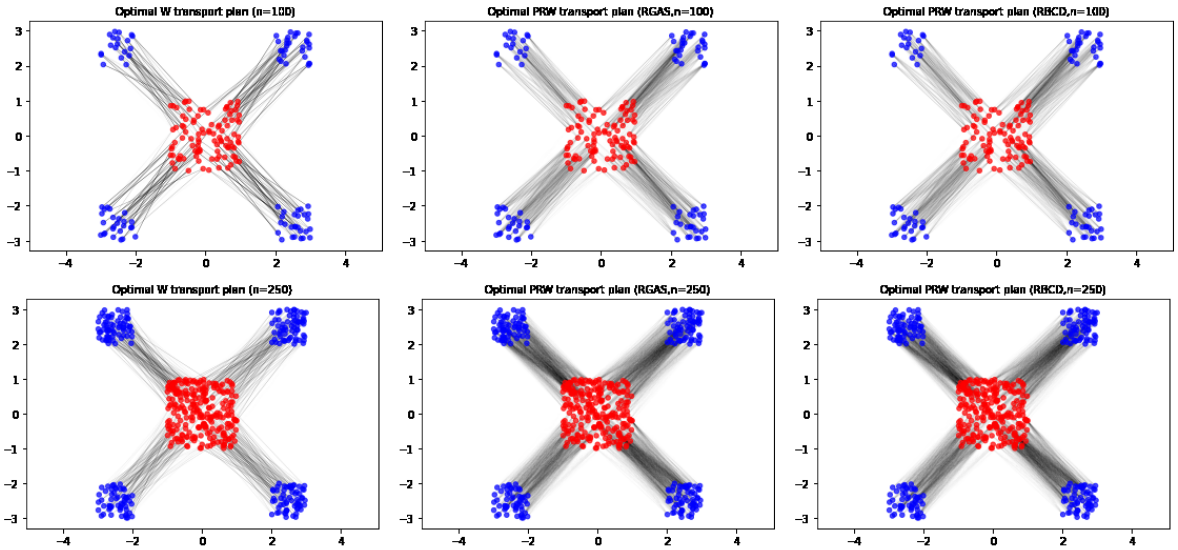

Figure 3. Fragmented hypercube with $(n, d) = (100, 30)$ (above) and $(n, d) = (250, 30)$ (bottom). Optimal transport plan obtained by the Wasserstein distance (left), the PRW distance calculated by the RGAS algorithm (middle) and the PRW distance calculated by the RBCD algorithm (right). Geodesics in the PRW space are robust to statistical noise.

$k \geq k^*$. Moreover, in this case we have $U^* \in \text{St}(d, k^*)$ with $U^*(1 : k^*, 1 : k^*) = I_{k^*}$. For all experiments in this subsection, we set the parameters as $\eta = 0.2$, $\epsilon_{RGAS} = \epsilon_1 = \epsilon_2 = 0.1, \tau_{RGAS} = \tau_{RBCD}/\eta$ and $\tau_{RBCD} = 0.005$. Figure 1 shows the computation of $\mathcal{P}_k^2(\hat{\mu}, \hat{\nu})$ on different k with $k^* \in \{2, 4, 7, 10\}$. After setting $n = 100, d = 30$ and generating the Fragmented Hypercube data with different k^*, we run both the RBCD and the RGAS (Lin et al., 2020a) algorithms for calculating the PRW distance. We see that the PRW value $\mathcal{P}_k^2(\hat{\mu}, \hat{\nu})$ grows more slowly after $k = k^*$ for both algorithms, which is reasonable since the last $d - k^*$ dimensions only represent noise. Furthermore, $\mathcal{P}_k^2(\hat{\mu}, \hat{\nu}) \approx 4k^*$ holds when $k = k^*$. Finally, we see that the solutions of both the RBCD and the RGAS algorithms achieve almost the same quality.

Table 1. CPU time for calculating PRW of the fragmented hypercube problem. We set $n = 100$.

DIMENSION d	20	50	100	250	500
RBCD	**0.14**	**0.20**	**0.39**	**1.70**	**4.41**
RGAS	0.37	0.42	0.66	1.92	4.55
RABCD	**0.10**	**0.09**	**0.16**	**0.77**	**3.14**
RAGAS	0.27	0.23	0.23	0.85	3.20
SRW(FW)	1.42	1.82	2.71	8.88	24.25

We present in Figure 2 the mean estimation error (MEE) for the sampled PRW distance $\mathcal{P}_k^2(\hat{\mu}, \hat{\nu})$ and the sampled Wasserstein distance $\mathcal{W}^2(\hat{\mu}, \hat{\nu})$ for different choices of $n \in \{25, 50, 100, 250, 500, 1000\}$. Theoretically, the MEE of both the PRW distance and the Wasserstein distance decreases as n increases. However, (Niles-Weed & Rigollet, 2019) showed that for spiked transport model the convergence rates of the estimation error are different. Specifically, when $d > 4$, we have for the sampled Wasserstein distance, $\mathbb{E}|\mathcal{W}(\mu, \nu) - \mathcal{W}(\hat{\mu}, \hat{\nu})| = O(n^{-1/d})$, and for the PRW distance, $\mathbb{E}|\mathcal{W}(\mu, \nu) - \mathcal{P}_k(\hat{\mu}, \hat{\nu})| = O(n^{-1/k})$, which significantly alleviate the curse of dimensionality because $k \ll d$. We set $k^* = 2, d = 20$ and generate $(\hat{\mu}, \hat{\nu})$ from (μ, ν) with n points. We calculate the estimation error in each run as $MEE = \mathbb{E}|\mathcal{P}_k^2(\hat{\mu}, \hat{\nu}) - 4k^*|$ for the PRW distance and $MEE = \mathbb{E}|\mathcal{W}^2(\hat{\mu}, \hat{\nu}) - 4k^*|$ for the Wasserstein distance. For the PRW distance, we further show the mean subspace

Table 2. CPU time for calculating PRW of the fragmented hypercube problem. We set $d = 50$.

n	50	100	250	500	1000
RBCD	**0.18**	**0.18**	**0.50**	**1.83**	**8.51**
RGAS	0.33	0.40	1.13	2.90	10.25
RABCD	**0.08**	**0.09**	**0.23**	**0.81**	**3.85**
RAGAS	0.17	0.21	0.61	1.48	5.39
SRW(FW)	1.24	1.81	4.58	15.42	64.65

Table 3. CPU time for calculating PRW of the fragmented hypercube problem. We set $n = d$.

DIMENSION d	20	50	100	250	500
RBCD	**0.06**	**0.16**	**0.35**	**2.62**	**12.75**
RGAS	0.18	0.30	0.61	3.20	13.12
RABCD	**0.06**	**0.08**	**0.12**	**1.14**	**7.97**
RAGAS	0.16	0.16	0.21	1.40	8.22
SRW(FW)	0.56	1.32	2.84	14.09	50.72

Table 4. CPU time for calculating PRW of the fragmented hypercube problem. We set $n = 10d$.

DIMENSION d	10	20	50	100	250
RBCD	**0.16**	**0.45**	**1.92**	**11.97**	**354.91**
RGAS	0.66	1.75	4.66	16.58	427.24
RABCD	**0.18**	**0.35**	**0.92**	**4.35**	**129.90**
RAGAS	0.79	1.26	2.35	7.22	157.07
SRW(FW)	1.86	3.88	18.47	90.83	1355.86

estimation error $\|\hat{\Omega} - \Omega^*\|_F$ in Figure 2. The subspace projection can be calculated as $\hat{\Omega} = \hat{U}\hat{U}^\top$, where $\hat{U}$ is the output of the algorithm. From Figure 2 we see that as n increases, both the MEE and the mean subspace estimation error decrease for both RBCD and RGAS algorithms. Moreover, we see that Wasserstein distance estimation behaves much worse than the PRW distance when the same number of samples are used.

We also plot the optimal transport plans between $(\hat{\mu}, \hat{\nu})$ generated by the Wasserstein distance and the PRW distance calculated by the RGAS and RBCD algorithms. The results are shown in Figure 3, where we considered the case when $k^* = 2, d = 30$ and $n \in \{100, 250\}$. From Figure 3 we see that in both cases, our RBCD algorithm can generate almost the same transport plans as the RGAS algorithm, which are also similar to the transportation plan generated by the Wasserstein distance.

Computational Time Comparison. We compare the computational time of five different algorithms on computing the PRW distance for the fragmented hypercube synthetic dataset mentioned above. The algorithms are: the Frank-Wolfe (FW) algorithm for computing the SRW distance (Paty & Cuturi, 2019), the RGAS and the RAGAS algorithms proposed in (Lin et al., 2020a), and our RBCD and RABCD algorithms, which is a variant of RBCD introduced in the supplementary material. The RGAS and the RAGAS algorithms are terminated when the stopping criteria (19) and (71) are satisfied, respectively. The RGAS and the RAGAS algorithms are terminated when $\|\text{grad}p(U^t)\|_F \leq \epsilon$. The FW algorithm is terminated when $\|\Omega^t - \Omega^{t-1}\|_F < (\epsilon \cdot \tau_{RBCD})^2$.

We fix $k^* = k = 2$, and generate the Fragmented Hypercube with varying n, d. We further set the thresholds $\epsilon_{RGAS} = \epsilon_{RAGAS} = \epsilon_1 = \epsilon = 0.1$ and $\epsilon_2 = \epsilon_1^2$. We set $\eta = 0.2$ when $d < 250$ and $\eta = 0.5$ otherwise. For fair comparison, we set the step size $\tau_{RGAS} = \tau_{RAGAS} = \tau_{RBCD}/\eta = \tau_{RABCD}/\eta$ and $\tau_{RBCD} = 0.001$. Tables 1 - 4 show the computational time comparison for different algorithms with different (n, d) pairs. All the reported CPU times are in seconds.We run each n, d pair for 50 times and take the average. From Tables 1 - 4, we see that our RBCD algorithm runs faster than the RGAS algorithm and our RABCD algorithm runs faster than the RAGAS algorithm in all cases. Moreover, we found that the advantage of RBCD (resp. RABCD) over RGAS (resp. RAGAS) is more significant when n is relatively larger than d. Moreover, the four algorithms for the PRW model are faster than the FW algorithm for computing the SRW distance.

4.2. Real Datasets

In this section, we conduct experiments on two real datasets. The first one is a dataset with movie scripts that was used in (Paty & Cuturi, 2019; Lin et al., 2020a). More specifically, we first compute the PRW distances between each pair of movies in a corpus of seven movie scripts (Paty & Cuturi, 2019; Lin et al., 2020a), where each script is transformed into a list of words. We then use word2vec (Mikolov et al., 2018) to transform each script into a measure over $\mathbb{R}^{300}$ with the weights corresponding to the frequency of the words. We then compute the PRW distances between a preprocessed corpus of six Shakespeare operas. For both experiments, we set the parameters as $\eta = 0.1, \tau_{RBCD} = 0.1, \epsilon = 0.001, \tau_{RGAS} = \tau_{RBCD}/\eta$ and project each point onto a 2-dimensional subspace. We run each experiments for 10 times and take the average running time. In Tables 5 and 6, the upper right half is the running time in seconds for RGAS/RBCD algorithms and the bottom left half is the $\mathcal{P}_k^2$ distance calculated by RGAS/RBCD algorithms. We highlight the smaller computational time in each upper right entry and the minimum PRW distance in each bottom left row. We see that the PRW distances are consistent and the RBCD algorithm runs faster than the RGAS algorithm in almost all cases. Moreover, the results indeed provide very useful information about the datasets. For example, from Table 5 we know that the movies “Dunkirk” and “Titanic” are close, and “Kill Bill Vol.1” and “Kill Bill Vol. 2” are close, because their PRW distances are small.

We then conduct further experiments on the MNIST dataset. Specifically, we extract the 128-dimensional features of each digit from a pre-trained convolutional neural network, which achieves an accuracy of 98.6% on the test set. Our task here is to compute the PRW distance by the RGAS and RBCD algorithms. We set parameters as $\eta = 8, \tau_{RBCD} = 0.004$ and $\tau_{RGAS} = \tau_{RBCD}/\eta$, $\epsilon = 0.1$ and compute the 2-dimensional projection distances between each pair of digits. All the distances are divided by 1000. We run the experiments for 10 times and take the average running time. In Table 7, the upper right half is the running time in seconds for RGAS/RBCD algorithms and the bottom left half is the $\mathcal{P}_k^2$ distance calculated by RGAS/RBCD algorithms. We highlight the smaller computational time in each upper right entry and the minimum PRW distance in each bottom

Table 5. Each entry of the **Bottom Left half** is the $\mathcal{P}_k^2$ distance calculated by RGAS/RBCD algorithms between different movie scripts. Each entry of the **Upper Right half** is the running time in seconds for RGAS/RBCD algorithms between different movie scripts. D = Dunkirk, G = Gravity, I = Interstellar, KB1 = Kill Bill Vol.1, KB2 = Kill Bill Vol.2, TM = The Martian, T = Titanic.

	D	G	I	KB1	KB2	TM	T
D	-/-	7.13/**6.01**	**8.64**/9.03	6.15/**5.52**	8.69/**7.99**	7.62/**6.60**	11.05/**10.24**
G	0.129/0.129	-/-	14.79/**12.68**	7.15/**5.95**	8.48/**7.13**	13.42/**11.06**	18.36/**16.24**
I	0.135/0.135	0.102/0.102	-/-	37.98/**32.06**	9.47/**7.99**	17.46/**14.80**	54.54/**49.46**
KB1	0.151/0.151	0.146/0.146	0.195/0.155	-/-	7.83/**6.87**	10.47/**8.91**	30.83/**21.55**
KB2	0.161/0.161	0.157/0.157	0.166/0.166	**0.088/0.088**	-/-	9.69/**8.47**	11.25/**9.23**
TM	0.137/0.137	**0.098/0.098**	**0.099/0.099**	0.146/0.146	0.152/0.152	-/-	27.15/**25.13**
T	**0.103/0.103**	0.128/0.128	0.135/0.135	0.136/0.136	0.138/0.138	0.134/0.134	-/-

Table 6. Each entry of the **Bottom Left half** is the $\mathcal{P}_k^2$ distance calculated by RGAS/RBCD algorithms between different Shakespeare plays. Each entry of the **Upper Right half** is the running time in seconds for RGAS/RBCD algorithms between different Shakespeare plays. H5 = Henry V, H = Hamlet, JC = Julius Caesar, TMV = The Merchant of Venice, O = Othello, RJ = Romeo and Juliet. (Note that the PRW distances are different from those reported in (Lin et al., 2020a). This is because we use a smaller η.)

	H5	H	JC	TMV	O	RJ
H5	-/-	56.5/**44.48**	6.63/**4.81**	19.87/**15.69**	25.91/**20.13**	14.06/**4.96**
H	0.123/0.123	-/-	18.97/**15.19**	22.11/**20.54**	14.65/**9.22**	**17.53**/20.34
JC	**0.117/0.117**	0.127/0.126	-/-	5.67/**4.72**	6.92/**5.35**	4.35/**4.10**
TMV	0.134/0.134	0.112/0.112	0.094/0.093	-/-	8.43/**6.65**	13.75/**10.67**
O	0.125/ 0.124	**0.091/ 0.091**	**0.086/ 0.086**	**0.090/0.090**	-/-	4.88/**4.17**
RJ	0.239/0.239	0.249/0.249	0.172/0.172	0.226/0.226	0.185/0.185	-/-

left row. We again observe that the PRW distances are consistent and the RBCD algorithm runs faster than the RGAS algorithm in almost all cases. Moreover, results in Table 7 also show some useful information that is consistent with our intuitation. For example, it indicates that digits 1 and 7 are easily confused, and digits 5 and 6 are easily confused, because they have small PRW distances.

More numerical results are provided in the supplementary materials.

Remark 4.1 *In our numerical experiments, we found that both RBCD and RGAS are sensitive to parameter η. This phenomenon was also observed when the Sinkhorn's algorithm was applied to solve the* REGOT *problem (Cuturi, 2013). Roughly speaking, if η is too small, then it may cause numerical instability, and if η is too large, then the solution to* REGOT *is far away from the solution to the original OT problem. Moreover, the adaptive algorithms RABCD and RAGAS are also sensitive to the step size τ, though they are usually faster than their non-adative versions RBCD and RGAS. We have tried our best to tune these parameters during our experiments so that the best performance is achieved for each algorithm. How to tune these parameters more systematically is left as a future work.*

5. Conclusion

In this paper, we have proposed RBCD and RABCD algorithms for computing the projection robust Wasserstein distance. Our algorithms are based on a novel reformulation of the regularized OT problem. We have analyzed the iteration complexity of both RBCD and RABCD algorithms, and this kind of complexity result seems to be new for BCD algorithm on Riemannian manifolds. Moreover, the complexity of arithmetic operations of our RBCD and RABCD algorithms is significantly better than that of the RGAS and RAGAS algorithms. We have conducted extensive numerical experiments and the results showed that our methods are more efficient than existing methods. Future work includes better tuning strategies of some parameters used in the algorithms.

Acknowledgements

We would like to thank the AC and all anonymous reviewers for very constructive suggestions that improved the quality of the paper. We are also grateful to Tianyi Lin for discussions on this topic. This research was partially supported by NSF HDR TRIPODS grant CCF-1934568, NSF grants CCF-1717943, CNS-1824553, CCF-1908258, DMS-1953210, ECCS-2000415 and CCF-2007797, and UC Davis CeDAR (Center for Data Science and Artificial Intelligence Research) Innovative Data Science Seed Funding Program.

Table 7. Each entry of the **Bottom Left half** is the $\mathcal{P}_k^2$ distance calculated by RGAS/RBCD algorithms for different pair of digits in the MNIST dataset. Each entry of the **Upper Right half** is the running time in seconds for RGAS/RBCD algorithms for different pair of digits in the MNIST dataset. (Note that the PRW distances are different from those reported in (Lin et al., 2020a). This is because we use a different stopping criteria and our results are more accurate.)

	D0	D1	D2	D3	D4	D5	D6	D7	D8	D9
D0	-/-	15.50/**13.64**	24.74/ **23.82**	12.95/**8.91**	21.91/**7.05**	11.50/**6.99**	15.66/**9.49**	**12.93**/17.29	14.82/**12.36**	12.30/**8.19**
D1	0.98/0.98	-/-	**21.70**/30.00	30.09/**20.91**	17.09/**13.72**	31.06/**30.21**	**31.31**/37.00	45.75/**29.92**	46.56/**44.88**	20.12/**18.19**
D2	**0.80/ 0.80**	**0.67/ 0.66**	-/-	**24.56**/35.84	26.15/**7.78**	13.28/**8.58**	20.43/**12.54**	22.89/**9.40**	23.78/**18.52**	12.55/**8.19**
D3	1.21/1.21	0.87/0.87	0.73/0.72	-/-	28.42/**18.37**	15.81/**11.74**	13.57/ **9.77**	14.08/**9.94**	17.01/**15.09**	32.50/**19.92**
D4	1.24/1.24	0.67/0.67	1.09/1.09	1.21/1.21	-/-	14.01/**11.15**	28.69/**13.04**	18.45/**12.14**	13.07/**7.77**	31.79/**22.33**
D5	1.04/1.04	0.85/0.85	1.09/1.09	**0.59/ 0.59**	1.01/1.01	-/-	14.40/**13.54**	19.82/**9.33**	20.92/**13.51**	18.58/**13.83**
D6	0.81/0.81	0.80/0.80	0.91/0.91	1.24/1.24	0.85/0.85	**0.72/ 0.72**	-/-	13.89/**11.11**	12.75/**8.46**	14.14/**8.91**
D7	0.86/0.85	**0.57/ 0.58**	0.70/0.71	0.73/0.73	0.80/0.80	0.92/0.92	1.11/1.11	-/-	12.67/**7.43**	28.14/**17.75**
D8	1.06/1.06	0.88/0.88	**0.68/ 0.68**	0.89/0.89	1.10/1.10	0.72/0.72	0.92/0.92	1.08/1.08	-/-	30.87/**10.15**
D9	1.09/1.09	0.86/0.86	1.07/1.07	0.84/0.84	**0.50/ 0.50**	0.78/0.78	1.11/1.11	0.61/0.61	0.87/0.87	-/-

References

Absil, P.-A., Mahony, R., and Sepulchre, R. *Optimization algorithms on matrix manifolds*. Princeton University Press, 2009.

Altschuler, J., Niles-Weed, J., and Rigollet, P. Near-linear time approximation algorithms for optimal transport via Sinkhorn iteration. In *Advances in neural information processing systems*, pp. 1964–1974, 2017.

Bellemare, M. G., Dabney, W., and Munos, R. A distributional perspective on reinforcement learning. In *International Conference on Machine Learning*, pp. 449–458, 2017.

Bonneel, N., Rabin, J., Peyré, G., and Pfister, H. Sliced and Radon Wasserstein barycenters of measures. *Journal of Mathematical Imaging and Vision*, 51(1):22–45, 2015.

Boumal, N., Absil, P.-A., and Cartis, C. Global rates of convergence for nonconvex optimization on manifolds. *IMA Journal of Numerical Analysis*, 39(1):1–33, 2019.

Brenier, Y. Polar factorization and monotone rearrangement of vector-valued functions. *Communications on pure and applied mathematics*, 44(4):375–417, 1991.

Chen, S., Ma, S., So, A. M.-C., and Zhang, T. Proximal gradient method for nonsmooth optimization over the Stiefel manifold. *SIAM Journal on Optimization*, 30(1): 210–239, 2020.

Cuturi, M. Sinkhorn distances: Lightspeed computation of optimal transport. In *Advances in neural information processing systems*, pp. 2292–2300, 2013.

Deshpande, I., Hu, Y.-T., Sun, R., Pyrros, A., Siddiqui, N., Koyejo, S., Zhao, Z., Forsyth, D., and Schwing, A. G. Max-sliced Wasserstein distance and its use for GANs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 10648–10656, 2019.

Duchi, J., Hazan, E., and Singer, Y. Adaptive subgradient methods for online learning and stochastic optimization. *JMLR*, 12:2121–2159, 2011.

Dudley, R. M. The speed of mean Glivenko-Cantelli convergence. *The Annals of Mathematical Statistics*, 40(1): 40–50, 1969.

Dvurechensky, P., Gasnikov, A., and Kroshnin, A. Computational optimal transport: Complexity by accelerated gradient descent is better than by Sinkhorn's algorithm. In *International Conference on Machine Learning*, pp. 1367–1376. PMLR, 2018.

Fournier, N. and Guillin, A. On the rate of convergence in Wasserstein distance of the empirical measure. *Probability Theory and Related Fields*, 162(3-4):707–738, 2015.

Ho, N. and Nguyen, X. Convergence rates of parameter estimation for some weakly identifiable finite mixtures. *The Annals of Statistics*, 44(6):2726–2755, 2016.

Jiang, B., Ma, S., So, A. M.-C., and Zhang, S. Vector transport-free svrg with general retraction for riemannian optimization: Complexity analysis and practical implementation. *arXiv preprint arXiv:1705.09059*, 2017.

Kasai, H., Jawanpuria, P., and Mishra, B. Riemannian adaptive stochastic gradient algorithms on matrix manifolds. In *International Conference on Machine Learning*, pp. 3262–3271, 2019.

Kingma, D. P. and Ba, J. L. Adam: A method for stochastic optimization. In *ICLR*, 2015.

Kolouri, S., Zou, Y., and Rohde, G. K. Sliced Wasserstein kernels for probability distributions. In *Proceedings of* *the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5258–5267, 2016.

Kolouri, S., Nadjahi, K., Simsekli, U., Badeau, R., and Rohde, G. Generalized sliced Wasserstein distances. In *Advances in Neural Information Processing Systems*, pp. 261–272, 2019.

Lei, J. Convergence and concentration of empirical measures under Wasserstein distance in unbounded functional spaces. *Bernoulli*, 26(1):767–798, 2020.

Lin, T., Ho, N., and Jordan, M. On efficient optimal transport: An analysis of greedy and accelerated mirror descent algorithms. In *International Conference on Machine Learning*, pp. 3982–3991. PMLR, 2019.

Lin, T., Fan, C., Ho, N., Cuturi, M., and Jordan, M. Projection robust Wasserstein distance and Riemannian optimization. In *NeurIPS*, volume 33, 2020a.

Lin, T., Zheng, Z., Chen, E. Y., Cuturi, M., and Jordan, M. I. On projection robust optimal transport: Sample complexity and model misspecification. *arXiv preprint arXiv:2006.12301*, 2020b.

Mikolov, T., Grave, É., Bojanowski, P., Puhrsch, C., and Joulin, A. Advances in pre-training distributed word representations. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.

Nagaraj, D., Jain, P., and Netrapalli, P. SGD without replacement: Sharper rates for general smooth convex functions. In *International Conference on Machine Learning*, pp. 4703–4711, 2019.

Nguyen, K., Ho, N., Pham, T., and Bui, H. Distributional sliced-wasserstein and applications to generative modeling. In *ICLR*, 2021a.

Nguyen, K., Nguyen, S., Ho, N., Pham, T., and Bui, H. Improving relational regularized autoencoders with spherical sliced fused gromov wasserstein. In *ICLR*, 2021b.

Niles-Weed, J. and Rigollet, P. Estimation of Wasserstein distances in the spiked transport model. *arXiv preprint arXiv:1909.07513*, 2019.

Ozair, S., Lynch, C., Bengio, Y., Van den Oord, A., Levine, S., and Sermanet, P. Wasserstein dependency measure for representation learning. In *Advances in Neural Information Processing Systems*, pp. 15604–15614, 2019.

Paty, F.-P. and Cuturi, M. Subspace robust Wasserstein distances. In *International Conference on Machine Learning*, pp. 5072–5081, 2019.

Rabin, J., Peyré, G., Delon, J., and Bernot, M. Wasserstein barycenter and its application to texture mixing. In *International Conference on Scale Space and Variational Methods in Computer Vision*, pp. 435–446. Springer, 2011.

Villani, C. *Optimal transport: old and new*, volume 338. Springer Science & Business Media, 2008.

Weed, J. and Bach, F. Sharp asymptotic and finite-sample rates of convergence of empirical measures in Wasserstein distance. *Bernoulli*, 25(4A):2620–2648, 2019.