Tree-Sliced Approximation of Wasserstein Distances

Tam Le [1] **Makoto Yamada** [2 1] **Kenji Fukumizu** [3 1] **Marco Cuturi** [4 5]

Abstract

Optimal transport (OT) theory provides a useful set of tools to compare probability distributions. As a consequence, the field of OT is gaining traction and interest within the machine learning community. A few deficiencies usually associated with OT include its high computational complexity when comparing discrete measures, which is quadratic when approximating it through entropic regularization; or supercubic when solving it exactly. For some applications, the fact that OT distances are not usually negative definite also means that they cannot be used with usual Hilbertian tools. In this work, we consider a particular family of ground metrics, namely tree metrics, which yield negative definite OT metrics that can be computed in linear time. By averaging over randomly sampled tree metrics, we obtain a tree-sliced-Wasserstein distance. We illustrate that the proposed tree-sliced-Wasserstein distances compare favorably with other baselines on various benchmark datasets.

1. Introduction

Many tasks in machine learning involve the comparison of two probability distributions, or histograms. Several geometries in the statistics and machine learning literatures have been routinely used, such as the Kullback-Leibler divergence, Fisher information metric, the χ^2 distance, or the Hellinger distance, to name a few. Among them, the optimal transport (OT) geometry, also known as Wasserstein (Villani, 2008), Monge-Kantorovich (Kantorovich, 1942), or Earth Mover's (Rubner et al., 2000), has gained traction in the machine learning community. OT plays an increasingly important role in machine learning (Cuturi & Doucet, 2014; Solomon et al., 2014b; Frogner et al., 2015; Kusner et al., 2015; Genevay et al., 2016; Ho et al., 2017; Arjovsky et al., 2017; Adler & Lunz, 2018; Lee & Raginsky, 2018; Ambrogioni et al., 2018; Gao et al., 2018), statistics (Panaretos et al., 2016; Ebert et al., 2017), or computer graphics (Solomon et al., 2014a; 2015; Bonneel et al., 2016; Lavenant et al., 2018).

Related work: The naive computation of OT between two discrete measures involves solving a network flow problem whose best known complexity scales super cubically in the size of these measures (Burkard & Cela, 1999; Tarjan, 1997). There are two notable lines of work to reduce the time complexity of OT. (i) The first one is to leverage simple ground costs. For instance, if one uses the binary metric $\mathbb{1}_{x \neq z}$ between two points, the OT distance is equivalent to the total variation distance (Villani, 2003, p.7). When measures are supported on the line $\mathbb{R}$ and the cost c is a convex function f of the absolute difference $|x - z|$ between two points, namely for $x, z \in \mathbb{R}$ we have $c(x, z) = f(|x - z|)$, then the OT distance is equal to the integral of f evaluated on the absolute difference between the generalized quantile functions of these two probability distributions (Santambrogio, 2015, §2). Other simplifications include thresholding the ground cost distance (Pele & Werman, 2009) or considering for a ground cost the shortest-path metric on a graph (Peyré & Cuturi, 2019, §6) (ii) The second one is to use regularization to approximate solutions of OT problems, notably entropy (Cuturi, 2013), which results in a problem that can be solved using Sinkhorn iterations. Genevay et al. (2016) extended this approach to the semi-discrete and discrete OT problems using stochastic optimization. Different variants of Sinkhorn algorithm have been proposed recently (Altschuler et al., 2017; Dvurechensky et al., 2018) and speed-ups are obtained when the ground cost is the quadratic Euclidean distance (Altschuler et al., 2018a;b; Tenetov et al., 2018) or more generally the heat kernel on geometric domains (Solomon et al., 2015). The convergence of Sinkhorn algorithm has been considered in (Franklin & Lorenz, 1989; Linial et al., 2000; Kalantari et al., 2008; Altschuler et al., 2017; Dvurechensky et al., 2018).

Contributions: In this work, we follow the first line of work to provide fast algorithms to approximate OT. In particular, we consider *tree metrics* as ground costs for OT, to define the tree-Wasserstein distances. Then, we propose tree-sliced Wasserstein by averaging over randomly tree

[1]RIKEN Center for Advanced Intelligence Project, Japan [2]Kyoto University, Japan [3]The Institute of Statistical Mathematics, Japan [4]Google Brain, Paris, France [5]CREST - ENSAE, France. Correspondence to: Tam Le <tam.le@riken.jp>.

Preprint. Work in progress.

metrics. We derive linear time algorithms for this class of OT with two practical families of tree metrics for both low-dimensional and high-dimensional data points. Moreover, this family of OT distances is also *negative definite*. Consequently, we propose positive definite kernels that use the OT geometry. Empirically, we show that averaging over random tree metrics for tree-sliced Wasserstein is essential for this proposed class of OT in applications.

2. Background

In this section, we briefly review definitions of optimal transport (OT) and tree metrics.

Let (Ω, d) be a measurable metric space. For any $x \in \Omega$, δ_x is the Dirac unit mass on x. Define the probability simplex $\mathbb{S}_n = \left\{u \in \mathbb{R}_+^n \mid u^T \mathbf{1}_n = 1\right\}$, where $\mathbf{1}_n$ is the n-dimensional vector of ones.

Optimal transport. In this work, we only consider either point clouds of the same finite cardinality or empirical measures with a finite number of supports for OT.

Point clouds. Let $X = (x_1, x_2, \ldots, x_n)$ and $Z = (z_1, z_2, \ldots, z_n)$ be two families of points of cardinality n, where $x_i, z_i \in \Omega, \forall i \leq n$. Let c be a ground cost metric, the OT problem between X and Z is defined as

$$d_{\text{OT}}(X, Z) = \min_{\sigma \in \Sigma_n} \sum_{i=1}^{n} c(x_i, z_{\sigma(i)}), \tag{1}$$

where Σ_n is the set of all permutations of n elements.

Empirical measures. Consider two empirical measures $\mu = \sum_{i=1}^n a_i \delta_{x_i}$ and $\nu = \sum_{i=1}^m b_i \delta_{z_i}$, where $x_i, z_j \in \Omega$, $\forall i \leq n, \forall j \leq m$, $a = (a_1, a_2, \ldots, a_n) \in \mathbb{S}_n$, and $b = (b_1, b_2, \ldots, b_m) \in \mathbb{S}_m$. Let c be a ground cost metric, the OT between μ and ν is defined as

$$d_{\text{OT}}(\mu, \nu) = \min_{P \in U(a,b)} \sum_{i=1}^{n} \sum_{j=1}^{m} P_{ij} c(x_i, z_j), \tag{2}$$

where $U(a, b) = \left\{P \in \mathbb{R}_+^{n \times m} \mid P\mathbf{1}_m = a, P^T \mathbf{1}_n = b\right\}$ is the *transportation polytope* of $a \in \mathbb{S}_n$ and $b \in \mathbb{S}_m$.

Tree metrics. A metric $(\mathbb{X}, d_{\text{T}})$ is a *tree metric* if there is a tree $\mathcal{T}$ with non-negative edge lengths whose nodes contain $\mathbb{X}$ and such that for every $x, z \in \mathbb{X}$, we have $d_{\text{T}}(x, z)$ equals to the length of the path between x and z (Semple & Steel, 2003) (Chapter 7, p.145–182).

3. Tree-Wasserstein Distances (TW Distances)

We call an optimal transport distance a tree-Wasserstein distance if the ground cost is a tree metric.

The tree-Wasserstein distance is built upon the UniFrac method (Lozupone & Knight, 2005; Lozupone et al., 2007) in metagenomics community. We recall that the UniFrac method is used for comparing microbial communities by measuring the phylogenetic distance between sets of taxa in a phylogenetic tree as the fraction of the branch length of the tree that leads to descendants from either one environment or the other, but not both (Lozupone & Knight, 2005).

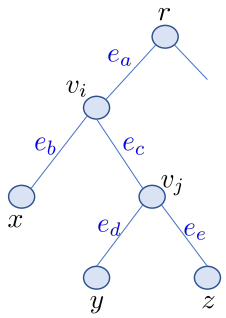

Figure 1. An illustration for a tree with root r. For node x on the lower left here, $\mathcal{P}(x) = \{e_a, e_b\}$. For an edge e_c, $h_{e_c}(x) = 0$, $h_{e_c}(y) = 1$. For a set $X = \{x, y\}$, $h_{e_a}(X) = 2$, $h_{e_c}(X) = 1$, and $h_{e_e}(X) = 0$.

In this section, we will leverage geometric structure of tree metrics on *local* spaces of OT, and rely on the optimal assignment formulation of OT (Equation 1) to derive a closed form for the tree-Wasserstein, which is similar as the UniFrac method (Lozupone et al., 2007). We further propose tree-sliced-Wasserstein distances by averaging over randomly sampled tree metrics, and use the tree-sliced-Wasserstein to build positive definite kernels on OT geometry.

Linear time computation for the TW distances: Let a tree metric d_{T} be the ground cost in the TW distance d_{TW} between $X = (x_1, x_2, \ldots, x_n)$ and $Z = (z_1, z_2, \ldots, z_n)$ in $\mathbb{P}_n$. Following the definition of tree metrics, there exists a tree $\mathcal{T}$ where $\forall i \leq n$, x_i, z_i are nodes of $\mathcal{T}$. Let r be a root of $\mathcal{T}$, and $\mathcal{P}(x)$ be the set of edges of the unique path from the root r to node x, $h_e(x)$ is 1 if the edge e appears in the path between the root r and x, and 0 otherwise. With a light abuse of notations, let $h_e(X) = \sum_{x \in X} h_e(x)$, as illustrated in Figure 1. We also denote $w(e)$ as a non-negative length (or weight) of an edge e in $\mathcal{T}$. Then, the tree-Wasserstein has a closed form as showed in Theorem 3.1.

Theorem 3.1. *Let $\mathbb{P}_n$ be a space of point clouds of cardinality n. Then, tree-Wasserstein distances between $X, Z \in \mathbb{P}_n$ has a following closed form:*

$$d_{TW}(X, Z) = \sum_{e \in \mathcal{T}} w(e) \, |h_e(X) - h_e(Z)|. \tag{3}$$

Proof. From the definition of tree metrics, $d_{\text{T}}(x, z)$ equals to the length of the path between x and z in $\mathcal{T}$, so we have

$$d_{\text{T}}(x, z) = \sum_{e \in \mathcal{P}(x)} w(e) \; + \sum_{e \in \mathcal{P}(z)} w(e) \; - 2 \sum_{e \in \mathcal{P}(x) \cap \mathcal{P}(z)} w(e).$$

Therefore, for any fixed $\sigma \in \Sigma_n$, consider the objective

function of Equation (1), and note that

$$\sum_{e\in\mathcal{P}(x)} w(e) = \sum_{e\in\mathcal{T}} w(e)h_e(x).$$

Then, we have:

$$\sum_{i=1}^{n} d_{\mathrm{T}}(x_i, z_{\sigma(i)}) = \sum_{\substack{e\in\mathcal{T}\\ x\in X}} w(e)h_e(x) + \sum_{\substack{e\in\mathcal{T}\\ z\in Z}} w(e)h_e(z) - 2\sum_{e\in T} w(e) \sum_{i=1}^{n} \min\{h_e(x_i), h_e(z_{\sigma(i)})\}.$$

Additionally, we have

$$\sum_{i=1}^{n} \min\{h_e(x_i), h_e(z_{\sigma(i)})\} \leq \min\{h_e(X), h_e(Z)\}.$$

Moreover, let construct a greedy assignment $\bar{\sigma} \in \Sigma_n$ as follow: we run a bottom-up traversal for edges on tree $\mathcal{T}$ starting from edges which are farthest to the root r. For each edge e in this order of the bottom-up traversal in $\mathcal{T}$, if it exists $x_i \in X$ and $z_j \in Z$ where $e \in \mathcal{P}(x_i)$ and $e \in \mathcal{P}(z_j)$, then we set $\bar{\sigma}(i) = j$, and remove x_i, z_j in X, Z respectively. When the bottom-up traversal is completed, if $X \neq \emptyset$, one can arbitrarily match for the rest. Since for any node x in $\mathcal{T}$, there is a *unique* shortest path from the root r to x, and the greedy assignment $\bar{\sigma}$ is constructed by the bottom-up traversal for edges in $\mathcal{T}$, we have

$$\sum_{i=1}^{n} \min\{h_e(x_i), h_e(z_{\bar{\sigma}(i)})\} = \min\{h_e(X), h_e(Z)\}.$$

Furthermore, note that

$$\sum_{\substack{e\in\mathcal{P}(x)\\ x\in X}} w(e) = \sum_{e\in\mathcal{T}} w(e) \sum_{x\in X} h_e(x) = \sum_{e\in\mathcal{T}} w(e)h_e(X),$$

$$h_e(X)+h_e(Z)-2\min\{h_e(X), h_e(Z)\} = |h_e(X)-h_e(Z)|.$$

Thus, the TW distance can be computed as

$$d_{\mathrm{TW}}(X, Z) = \sum_{e\in\mathcal{T}} w(e)\,|h_e(X) - h_e(Z)|.$$

■

Moreover, let consider the smallest subtree $\tilde{\mathcal{T}}$ having the same root r in $\mathcal{T}$ and containing all $x_i \in X$ and $z_i \in Z$. Without any loss, one can remove nodes of degree 2, except the root node r, to construct a tree $\bar{\mathcal{T}}$ with root r from $\tilde{\mathcal{T}}$. $\bar{\mathcal{T}}$ has at most $4n$ nodes, consequently at most $(4n-1)$ edges. Therefore, from Equation (3), the TW distance can be computed in linear time $O(n)$. We summarize a computation of the TW distance for point sets of a cardinality n in $\mathbb{P}_n$ in Algorithm 1[1].

[1]In practice, one can obtain the path from root to each node x (i.e. $\mathcal{P}(x)$) along a tree construction procedure. Or, one can index those paths in a preprocessing phase by performing a bottom-up traversal over all edges in tree $\mathcal{T}$ in linear time with respect to the number of nodes in $\mathcal{T}$.

Algorithm 1 Compute d_{TW} for point clouds of cardinality n

Input: $X, Z \in \mathbb{P}_n$, a tree $\mathcal{T}$ corresponding to a ground tree metric in d_{TW}, and denote m as the number of edges in tree $\mathcal{T}$.

Output: $d_{\mathrm{TW}}(X, Z)$

1: Compute a count vector $u \in \mathbb{R}_+^m$ for X where each coordinate of u corresponds to an edge e in $\mathcal{T}$, and equals to $h_e(X)$.
2: Compute a count vector $v \in \mathbb{R}_+^m$ for Z similarly.
3: Compute a weighted vector $\bar{w} \in \mathbb{R}_+^m$ where each coordinate of $\bar{w}$ corresponds to an edge e in $\mathcal{T}$, and equals to $w(e)$.
4: Compute $d_{\mathrm{TW}}(X, Z) = \bar{w}^T\,|u - v|$, where $|\cdot|$ is an element-wise absolute operator.

Remark 1. *Evans & Matsen (2012) showed a relation between the UniFrac and OT distance by relying on the Kantorovich duality of OT and total mass of phylogenetic subtrees. We note that this result was also implicitly showed in (Do Ba et al., 2011) based on network flow and total mass of subtrees. Later, it was noted in (McGregor & Stubbs, 2013), also based on total mass of subtrees. Differently, in our work, we derive the TW distances based on general tree metrics, and optimal assignment formulation of OT (Equation (1)).*

Remark 2. *In literature, there are a few more work related to our proposed class of OT with tree metrics (Kloeckner, 2015; Sommerfeld & Munk, 2018). In particular, Kloeckner (2015) studied geometric properties of OT space for measures on an ultrametric space, and Sommerfeld & Munk (2018) focused on statistical inference for empirical OT on finite spaces including tree metrics.*

Negative definiteness for the TW distances:

Theorem 3.2. *Let $\mathbb{P}_n$ be a space of point clouds of cardinality n. Then, the tree-Wasserstein distance d_{TW} is negative definite on $\mathbb{P}_n$.*

Proof. Let m be the number of edges in tree $\mathcal{T}$. From Equation (3), $h_e(X)$ with $e \in \mathcal{T}$ can be considered as a feature map for point cloud X from $\mathbb{P}_n$ to $\{0, 1, \ldots, n\}^m$. Consequently, tree-Wasserstein distance is equivalent to a weighted L_1 distance, with positive weights $w(e)$, between these feature maps. Therefore, tree-Wasserstein is negative definite. ■

Positive definite tree-Wasserstein kernels.

Lemma 3.3. *Let $\mathbb{P}_n$ be a space of point sets of cardinality n. Given $t > 0$, $X, Z \in \mathbb{P}_n$, we proposed tree-Wasserstein kernels, defined as $k_{TW}(X, Z) = \exp(-t d_{TW}(X, Z))$. Then, the tree-Wasserstein kernels are positive definite on $\mathbb{P}_n$.*

Proof. From Theorem 3.2, d_{TW} is negative definite on $\mathbb{P}_n$. Following (Berg et al., 1984) (Theorem 3.2.2, p.74), the proposed TW kernels k_{TW} are positive definite on $\mathbb{P}_n$. ■

Practical families of tree metrics for general cases[2].

• **Partition-based tree metrics:** For low-dimensional *local* spaces, one can construct a partition-based tree metric with a tree structure $\mathcal{T}$ as follows:

For simplicity, we use $\mathbb{R}^2$ as a running example. Assume that data points are in a square region of $\mathbb{R}^2$ with side $\lambda/2$. We then randomly expand the square region into a bigger one with side at most λ. Inspired by Indyk & Thaper (2003), we use a following recursive procedure to construct a tree $\mathcal{T}$ from the square region with side at most λ: for each square region $\mathtt{s}$ with side ℓ, there are three cases: (i) if $\mathtt{s}$ does not contain any data points, we discard it, (ii) if $\mathtt{s}$ contains 1 data point, we use either this data point or the center of $\mathtt{s}$ as a node in $\mathcal{T}$, and (iii) if $\mathtt{s}$ contains more than 1 data point, we use its center to represent it as a node x in tree $\mathcal{T}$, then equally partition $\mathtt{s}$ into 4 smaller child square regions with side $\ell/2$ to obtain potential child nodes of x in $\mathcal{T}$, then we recursive the procedure for those child square regions with side $\ell/2$. One can use any metrics in $\mathbb{R}^2$ to obtain a length for each edge in $\mathcal{T}$.

To reduce a quantization problem for the partition-based approach in d_{TW}, we use several different partition-based tree metrics, obtained by using different randomly expansions for the original square region, to compute several corresponding d_{TW}, then we average them, namely the *tree-sliced-Wasserstein* distance.

• **Clustering-based tree metrics:** For high-dimensional *local* spaces, the number of partitioned regions, in the recursive procedure of *partition-based* tree metrics, grows exponentially with respect to the number of dimensions of *local* spaces. To overcome this high-dimension problem, we leverage a distribution of data points to adaptively partition a space via clustering, inspired by the clustering-based approach for a space subdivision in Improved Fast Gauss Transform (Yang et al., 2005; Morariu et al., 2009). We derive a similar recursive procedure as in partition-based tree metrics, but use the farthest clustering of Gonzalez (1985) to partition for data points. The complexity of the farthest clustering into κ clusters for n data points is $O(n \log \kappa)$ by using Feder & Greene algorithm (1988). So, the complexity to build a clustering-based tree metric is linear with respect to n.

Similarly, one can consider its corresponding *tree-sliced-Wasserstein* distance, where the TW distances are averaged over a several times of computations with different clustering-based tree metrics, obtained by using different initializations for the farthest clustering of Gonzalez (1985).

Remark 3. *In practice, one can use a predefined highest level of tree as a stopping condition for a tree construction procedure. We note that the shortest path from the root to each node can be obtained along the tree construction procedure. Moreover, averaging over randomly tree metrics in tree-sliced-Wasserstein distances is necessary for applications as illustrated in our experiments.*

A relation between the TW distance and OT with Euclidean ground metric: Given a modified partition-based tree metric d_{T}^H, where a corresponding tree $\mathcal{T}$ is constructed as follow: assume that all data points are in a m-dimensional hypercube with side λ. At a height level 0 in $\mathcal{T}$, there is 1 hypercube with side λ. For each hypercube with side ℓ, at a height level i in $\mathcal{T}$, we partition it into 2^m child hypercubes with side $\ell/2$, corresponding to 2^m child nodes in $\mathcal{T}$ at a height level $(i+1)$. Therefore, at a height level i in $\mathcal{T}$, there are $(2^i)^m$ hypercubes with side $\lambda/2^i$. Assume that at a height level H in $\mathcal{T}$, all data points are separated into different hypercubes, then the partitioning procedure is stopped at that level. Each hypercube is represented by its center as a node in $\mathcal{T}$, and each edge in $\mathcal{T}$ is computed by Euclidean distance.

Let consider the OT with Euclidean ground metric, denoted as $d_{\text{OT}}(\cdot, \cdot; L_2)$, and the TW distance with the partition-based tree metric d_{T}^H, denoted as $d_{\text{TW}}(\cdot, \cdot; d_{\text{T}}^H)$, for $X = (x_1, x_2, \ldots, x_n)$ and $Z = (z_1, z_2, \ldots, z_n)$ in $\mathbb{P}_n$. We show a relation between $d_{\text{OT}}(\cdot, \cdot; L_2)$ and $d_{\text{TW}}(\cdot, \cdot; d_{\text{T}}^H)$ as in Lemma 3.4.

Lemma 3.4. *Let $X, Z \in \mathbb{P}_n$, λ be a side of a m-dimensional hypercube containing all data points in X and Z, and H be the height level of tree $\mathcal{T}$ for d_T^H, then $d_{OT}(X, Z; L_2) \leq d_{TW}(X, Z; d_T^H)/2 + \lambda n \sqrt{m}/2^H$.*

Proof. At a height level i in $\mathcal{T}$, the maximum Euclidean distance between any two data points in a same hypercube, denoted as Δ_i, equals to $\lambda\sqrt{m}/2^i$. Let $E_{i(i+1)}$ be a set of all edges between a height level i and a height level $(i+1)$ in $\mathcal{T}$. So, for any $e \in E_{i(i+1)}$, $w(e) = \lambda\sqrt{m}/2^{i+1}$. Let q_i be the number of matched pairs at a height level i. So, $(q_i - q_{i+1})$ is the number of pairs matched at a height level i, but unmatched at a height level $(i+1)$. Moreover, observe that the number of unmatched pairs at a height level i is $n - q_i = \frac{1}{2}\sum_{e \in E_{(i-1)i}} |h_e(X) - h_e(Z)|$. So, from Equation (3),

[2]General cases are meant that a tree structure is not known, and required to construct from data for computing the TW distances.

Table 1. Results on SVM classification for MPEG7 and Orbit datasets. For each dataset, two numbers in the parenthesis are corresponding to the number of persistence diagrams (PD) and the maximum number of points in PD respectively. For each kernel, the averaged accuracy (%) and standard deviation are shown on its first line, and time consumption (seconds) is shown in a parenthesis on its second line. For a trade-off between speed and performance, we set $(n_s = 6, H_{\mathcal{T}} = 6)$, and $(n_s = 12, H_{\mathcal{T}} = 5)$ for tree-sliced-Wasserstein distances in k_{TW} in MPEG7 and Orbit datasets respectively. We highlight our approach and its results in bold.

	k_{PSS}	k_{PWG}	k_{SW}	k_{PF}	k_{OT}	$\boldsymbol{k_{\text{TW}}}$
MPEG7	73.33 ± 4.17	74.83 ± 4.36	76.83 ± 3.75	80.00 ± 4.08	69.17 ± 5.40	$\mathbf{81.83 \pm 3.15}$
(200/80)	(7.51)	(5.23)	(1.55)	(6.63)	(7.98)	**(4.48)**
Orbit	72.38 ± 2.41	76.63 ± 0.66	83.60 ± 0.87	85.87 ± 0.77	77.57 ± 0.75	$\mathbf{86.31 \pm 1.01}$
(5K/300)	(11024)	(8756)	(6473)	(9891)	(433752)	**(1480)**

$d_{\text{TW}}(X, Z; d_{\text{T}}^H) = \sum_{i=0}^{H-1} 2w(e|e \in E_{i(i+1)})(n - q_{i+1}) = \sum_{i=0}^{H-1} \Delta_i(n - q_{i+1})$. Additionally, note that $q_0 = n$, $q_H = 0$, and $\Delta_i = \Delta_{i-1}/2$, then we have

$$
\begin{aligned}
d_{\text{OT}}(X, Z; d_{L_2}) &\leq \sum_{i=0}^{H-1} \Delta_i \left(q_i - q_{i+1}\right) \\
&= d_{\text{TW}}(X, Z; d_{\text{T}}^H) - \sum_{i=0}^{H-1} \Delta_i (n - q_i) \\
&= d_{\text{TW}}(X, Z; d_{\text{T}}^H) - \sum_{i=1}^{H} \Delta_{i-1} (n - q_i)/2 + n\Delta_H \\
&= d_{\text{TW}}(X, Z; d_{\text{T}}^H)/2 + \lambda n \sqrt{m}/2^H.
\end{aligned}
$$

■

Generalization for empirical measures. Let $\mathbb{M}$ be a space of empirical measures with a finite number of support. Consider a tree $\mathcal{T}$ corresponding to a ground tree metric in d_{TW}. For any $\mu = \sum_{i=1}^n a_i \gamma_{x_i}$ in $\mathbb{M}$ (Section 2), with a slight abuse of notations, we define

$$h_e(\mu) = \sum_{i=1}^n a_i h_e(x_i).$$

Moreover, empirical measures (e.g. $\mu = \sum_{i=1}^n a_i \gamma_{x_i}$) can be considered as a generalization of point clouds (e.g. $X = (x_1, x_2, \ldots, x_n)$) since a_i can be regarded as a frequency of appearances of x_i. Therefore, all results of the TW distances on $\mathbb{P}_n$ (i.e. Theorem 3.1, Theorem 3.2, Lemma 3.3 and Lemma 3.4) are also hold for the TW distances on $\mathbb{M}$.

Special cases of the TW distances. We highlight some special cases of the TW distances which are either trivial and/or have similar spirits to other different tools, considered in literature.

• **OT with a ground binary metric:** For a binary metric $\mathbb{1}_{x \neq z}$ between two points on a set $\mathbb{X}$, one can construct a tree $\mathcal{T}$ where all points in $\mathbb{X}$ are leaves, an additional virtual point is its root, and all edges in $\mathcal{T}$ have a same length $1/2$.

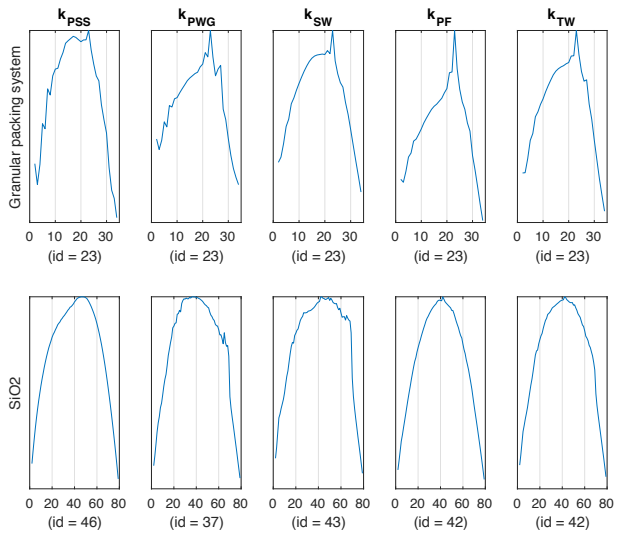

Figure 2. The kernel Fisher discriminant ratio (KFDR) graphs on granular packing system and SiO_2 datasets. For k_{TW}, tree-sliced-Wasserstein distances are computed with $(n_s = 12, H_{\mathcal{T}} = 6)$.

Then, $\mathbb{1}_{x \neq z}$ equals to the length of the path between x and z in $\mathcal{T}$. Hence, binary metric is a tree metric.

• **OT with ground metrics in 1-dimensional *local* spaces:** For a metric d in 1-dimensional *local* spaces of OT, all data points in the *local* spaces lay on a line which is a trivial case of trees. So, all data points are nodes in a tree, and a length of an edge equals to the distance d between two nodes of the edge. Thus, d is a tree metric.

• **OT with ground ultrametrics:** An ultrametric is also known as non-Archimedean metric or isosceles metric (Shkarin, 2004). Ultrametrics strengthen a triangle inequality to a *strong inequality* (i.e. for any x, y, z in an ultrametric space, $d(x, z) \leq \max\{d(x, y), d(y, z)\}$). Note that binary metrics are a special case of ultrametrics since binary metrics satisfy the *strong inequality*. Following (Johnson, 1967) (§.1, p.245–247), ultrametric implies a tree structure which can be constructed by hierarchical clustering schemes. Therefore, ultrametric is a tree metric. Furthermore, we note that ultrametrics have similar spirits with *strong kernels* and *hierarchy-induced kernels* which are key components to form valid optimal assignment kernels for applications with

Table 2. Results of k_{TW} with SVM on MPEG7 dataset over different parameters $(n_s, H_{\mathcal{T}})$ for tree-sliced-Wasserstein distances. For each pair $(n_s, H_{\mathcal{T}})$, the averaged accuracy (%) and standard deviation are shown on its first line while time consumption (seconds) is shown in a parenthesis on its second line.

	$n_s = 1$	$n_s = 6$	$n_s = 12$	$n_s = 18$	$n_s = 24$	$n_s = 30$
$H_{\mathcal{T}} = 3$	58.00 ± 3.75 (0.03)	71.33 ± 3.31 (0.15)	73.33 ± 2.83 (0.29)	76.17 ± 3.60 (0.52)	79.00 ± 4.17 (0.62)	77.83 ± 4.78 (0.83)
$H_{\mathcal{T}} = 4$	71.17 ± 3.69 (0.08)	79.17 ± 4.98 (0.44)	80.17 ± 2.54 (1.26)	79.50 ± 3.24 (1.68)	79.67 ± 4.18 (1.91)	83.00 ± 3.75 (2.57)
$H_{\mathcal{T}} = 5$	74.17 ± 3.36 (0.27)	79.17 ± 3.16 (1.68)	80.00 ± 4.39 (2.86)	80.33 ± 3.91 (3.97)	80.83 ± 3.95 (5.78)	80.83 ± 1.96 (7.32)
$H_{\mathcal{T}} = 6$	78.00 ± 3.22 (0.73)	81.83 ± 3.15 (4.48)	81.50 ± 2.58 (8.96)	80.67 ± 12.78 (12.78)	83.00 ± 2.92 (18.40)	81.00 ± 5.78 (22.86)

graph classification (Kriege et al., 2016).

4. Experimental Results

In this section, we apply our proposed TW kernel on topological data analysis and word embedding-based document classification.

4.1. Topological Data Analysis (TDA)

Topological data analysis (TDA have recently gained interest within the machine learning community (Reininghaus et al., 2015; Kusano et al., 2016; Carriere et al., 2017; Le & Yamada, 2018). TDA is a powerful tool for statistical analysis on geometric structured data such as linked twist maps, or material data. TDA employs algebraic topology methods, such as *persistence homology*, to extract robust topological features (i.e. connected components, rings, cavities) and output 2-dimensional point multisets, known as *persistence diagrams* (PD) (Edelsbrunner et al., 2000; Edelsbrunner & Harer, 2008). Each 2-dimensional point in PD summarizes a lifespan, corresponding to birth and death time as its coordinates, of a particular topological feature.

Setup. We evaluated our proposed TW kernel k_{TW} (Section 3) for orbit recognition and object shape classification with support vector machines (SVM), as well as change point detection for material data analysis with kernel Fisher discriminant ratio (KFDR) (Harchaoui et al., 2009). Generally, we used the same setting as in (Le & Yamada, 2018) for these TDA experiments[3]. We consider five baseline kernels for persistence diagrams (PD): (i) Persistence Scale Space kernel (k_{PSS}) (Reininghaus et al., 2015), (ii) Persistence Weighted Gaussian kernel (k_{PWG}) (Kusano et al., 2016), (iii) Sliced Wasserstein kernel (k_{SW}) (Carriere et al., 2017), (iv) Persistence Fisher kernel (k_{PF}) (Le & Yamada, 2018), and (v) optimal transport (OT)[4] kernel, defined as $k_{\text{OT}} = \exp(-t d_{\text{OT}})$ for $t > 0$, as in (Zhang et al., 2007; Cuturi, 2013). Since k_{OT} is indefinite, we regularize its corresponding kernel matrices by adding a sufficiently large diagonal term as in (Cuturi, 2013). We consider n_s randomized partition-based tree metrics, built with a predefined highest level $H_{\mathcal{T}}$ of tree $\mathcal{T}$ as a stopping condition, corresponding to a tree-sliced-Wasserstein distance with n_s tree-slices for d_{TW} in k_{TW}. We typically use a cross validation to choose hyper-parameters, and follow corresponding authors of those baseline kernels to form set of candidates. For k_{TW} and k_{OT}, we choose $1/t$ from $\{1, q_{10}, q_{20}, q_{50}\}$ where q_s is the $s\%$ quantile of a subset of TW distances and OT distances respectively, observed on a training set. We use one-vs-one strategy with Libsvm (Chang & Lin, 2011) for multi-class classification, $\{10^{-2:1:2}\}$ as a set of regularization candidates for SVM, and DIPHA toolbox[5] to extract PD.

Orbit recognition. Adams et al. (2017) (§6.4.1) proposed a synthesized dataset for link twist map, a discrete dynamical system to model flows in DNA microarrays (Hertzsch et al., 2007). Given an initial position $(a_0, b_0) \in [0, 1]^2$, and $t > 0$, an orbit is modeled as $a_{i+1} = a_i + t b_i(1 - b_i) \mod 1$, and $b_{i+1} = b_i + t a_{i+1}(1 - a_{i+1}) \mod 1$. There are 5 classes, corresponding to 5 different parameters $t = 2.5, 3.5, 4, 4.1, 4.3$. For each class, we generated 1000 orbits with random initial positions. Each orbit contains 1000 points. We consider 1-dimensional topological features for PD, extracted with Vietoris-Rips complex filtration (Edelsbrunner & Harer, 2008), and use a random split $70\%/30\%$ for training and test with 100 repeats for SVM. Results of SVM, and computational time are shown in the third row of Table 1. The proposed k_{TW} compares favorably with other baseline kernels. Moreover, k_{TW} and k_{PF} which enjoy OT geometry and Fisher information geometry respectively, *without either approximation or regularization* for PD, clearly outperform other approaches. Furthermore, the computational time of k_{TW} is less than other baseline kernels. Especially, k_{TW} is 293 times faster than k_{OT}. We also illustrate a trade-off of computational time and per-

[3] Le & Yamada (2018) kindly helped us evaluate k_{TW} and k_{OT} on their system for a fair comparison on TDA.

[4] We used a fast OT implementation (e.g. on MPEG7 dataset, it took 7.98 seconds while the mex-file with Rubner's implementation requires 28.72 seconds).

[5] https://github.com/DIPHA/dipha

Table 3. Results of k_{TW} with SVM on Orbit dataset over different parameters $(n_s, H_{\mathcal{T}})$ for tree-sliced-Wasserstein distances. For each pair $(n_s, H_{\mathcal{T}})$, the averaged accuracy (%) and standard deviation are shown on its first line while time consumption (seconds) is shown in a parenthesis on its second line.

	$n_s = 1$	$n_s = 6$	$n_s = 12$	$n_s = 18$	$n_s = 24$	$n_s = 30$
$H_{\mathcal{T}} = 3$	75.96 ± 0.67 (9)	83.12 ± 0.60 (53)	83.33 ± 0.80 (103)	83.84 ± 0.85 (150)	84.53 ± 0.83 (193)	83.56 ± 0.83 (244)
$H_{\mathcal{T}} = 4$	79.37 ± 0.86 (24)	84.55 ± 0.41 (143)	85.58 ± 1.01 (294)	85.37 ± 0.84 (456)	85.64 ± 0.65 (604)	85.72 ± 0.88 (799)
$H_{\mathcal{T}} = 5$	80.63 ± 1.00 (109)	85.34 ± 0.86 (685)	86.31 ± 1.01 (1480)	86.75 ± 0.60 (1943)	86.57 ± 0.71 (2607)	85.73 ± 0.76 (3670)
$H_{\mathcal{T}} = 6$	82.73 ± 0.90 (541)	85.62 ± 0.63 (3458)	86.27 ± 0.66 (7757)	86.62 ± 0.42 (11718)	85.77 ± 0.70 (19752)	85.73 ± 0.59 (22034)

Table 4. Computational time (seconds) for granular packing system ($35/20.4K$) and SiO_2 ($80/30K$) datasets. For each dataset, the first number in the parenthesis is the number of persistence diagrams (PD) while the second one is the maximum number of points in PD. For k_{TW}, tree-sliced-Wasserstein distances are computed with $(n_s = 12, H_{\mathcal{T}} = 6)$. Computation for OT is out of memory due to many points in PD. We highlight our approach and its computational time in bold.

	k_{PSS}	k_{PWG}	k_{SW}	k_{PF}	$\boldsymbol{k_{\text{TW}}}$
Granular	38.14	17.44	8.30	22.70	**2.01**
SiO_2	515	288	249	318	**6**

formance for $(n_s, H_{\mathcal{T}})$ in tree-sliced-Wasserstein distances for k_{TW} in Table 3. Averaging over randomly tree metrics as in our proposed tree-sliced-Wasserstein is essential in Orbit dataset since the results of k_{TW} with $n_s > 1$ clearly outperforms those with $n_s = 1$.

Object shape classification. We evaluated object shape classification on a 10-class subset of MPEG7 dataset (Latecki et al., 2000), containing 20 samples for each class as in (Le & Yamada, 2018). For simplicity, we use the same procedure as in (Le & Yamada, 2018) to extract 1-dimensional topological features for PD with Vietoris-Rips complex filtration[6](Edelsbrunner & Harer, 2008). We also used a random split $70\%/30\%$ for training and test with 100 repeats for SVM. Results of SVM and computational time are summarized in the second row of Table 1. The performances of the proposed k_{TW} are comparative with k_{PF}, and clearly outperform other baseline kernels. Therefore, geometry for PD plays an important role for kernel approaches in TDA. The computational time of k_{TW} is slower than k_{SW}, but faster than other baseline kernels. A trade-off of computational time and performance for $(n_s, H_{\mathcal{T}})$ in tree-sliced-Wasserstein distances for k_{TW} is shown in Table 2. Similarly, averaging over randomly tree metrics for MPEG7 dataset also helps to improve performances for k_{TW}.

Change point detection for material data analysis. We consider granular packing system (Francois et al., 2013) and SiO_2 (Nakamura et al., 2015) datasets for change point detection problem with KFDR as a statistical score. We extracted 2-dimensional topological features for PD in granular packing system dataset, and 1-dimensional topological features for PD in SiO_2 dataset, both with ball model filtration. Following (Kusano et al., 2018; Le & Yamada, 2018), the regularization parameter in KFDR is set to 10^{-3}. KFDR graphs for the granular packing system and SiO_2 datasets are shown in Figure 2. For granular tracking system dataset, all kernel approaches obtain the change point as the 23^{rd} index, which support an observation result (corresponding id = 23) in (Anonymous, 1972) . For SiO_2 dataset, results of all kernel methods are within a supported range ($35 \leq$ id ≤ 50), obtained by a traditional physical approach (Elliott, 1983). The KFDR results of k_{TW} compare favorably with those of other baseline kernels. As shown in Table 4, k_{TW} is faster than other baseline kernels. We note that we omit the baseline k_{OT} for this application since computation of OT is out of memory.

4.2. Word Embedding-based Document Classification

Kusner et al. (2015) proposed Word Mover's distances for document classification. Each document is regarded as an empirical measure where each word plays a role of a support, and its frequency is considered as a weight for the support. Kusner et al. (2015) used word embedding such as *word2vec* to map each word to a vector data point. Or, Word Mover's distances are optimal transport metrics between empirical measures (i.e. documents) where its ground cost is a metric on word embedding space.

Setup. We evaluated our proposed TW kernel on 4 datasets: `TWITTER`, `RECIPE`, `CLASSIC` and `AMAZON`, following the approach of Word Mover's distances (Kusner et al., 2015), for document classification with SVM. Statistical characteristics for those datasets are summarized in the first row of Table 6. We used the *word2vec* word embedding (Mikolov et al., 2013), pre-trained on Google News[7], which contains about 3 million words/phrases. *word2vec* maps words/phrases into a vector in $\mathbb{R}^{300}$. For all datasets, we removed all SMART stop word (Salton & Buckley, 1988),

[6]Turner et al. (2014) proposed a more complicated and advanced filtration for this task.

[7]https://code.google.com/p/word2vec

Table 5. Results of SVM with different parameters $(n_s, H_\mathcal{T})$ with $\kappa = 4$ for k_{TW} and different n_s for k_{TW} on TWITTER dataset.

	$n_s = 1$	$n_s = 5$	$n_s = 10$	$n_s = 20$	$n_s = 30$	$n_s = 50$
k_{TW} $(H_\mathcal{T} = 4, \kappa = 4)$	69.74 ± 0.41 (12)	71.68 ± 0.26 (56)	71.66 ± 0.50 (113)	72.28 ± 0.76 (239)	72.94 ± 0.80 (364)	72.79 ± 0.62 (605)
k_{TW} $(H_\mathcal{T} = 5, \kappa = 4)$	71.57 ± 0.71 (46)	72.67 ± 0.42 (231)	72.86 ± 0.78 (472)	73.24 ± 0.43 (939)	73.37 ± 0.35 (1469)	73.42 ± 0.42 (2400)
k_{TW} $(H_\mathcal{T} = 6, \kappa = 4)$	72.81 ± 0.65 (180)	72.82 ± 0.51 (929)	73.09 ± 0.37 (1914)	73.67 ± 0.63 (3703)	73.37 ± 0.39 (5569)	73.54 ± 0.81 (9166)
k_{SW}	68.86 ± 0.13 (152)	68.85 ± 0.11 (669)	68.89 ± 0.14 (1316)	70.10 ± 0.75 (2746)	69.61 ± 0.47 (4097)	70.04 ± 0.40 (6635)

and further dropped words in documents if they are not available in the pre-trained *word2vec* as in (Kusner et al., 2015). We used baseline kernels in the form of $\exp(-td)$ where d is a document distance and $t > 0$. We considered 2 baseline document distances based on Word Mover's: (i) OT with Euclidean ground metric (Kusner et al., 2015), and (ii) 1D-sliced-Wasserstein. With a light abuse of notations, we denote those baselines as k_{OT} and k_{SW} for their corresponding kernels respectively. We consider n_s randomized clustering-based tree metrics, built with a predefined highest level $H_\mathcal{T}$ of tree $\mathcal{T}$ as a stopping condition, corresponding to tree-sliced-Wasserstein distances with n_s tree-slices for d_{TW} in k_{TW}. We also used a same regularization (Section 4.1) for kernel k_{OT} matrices due to its indefiniteness. We randomly split each dataset into $70\%/30\%$ for training and test with 100 repeats for SVM. We follow the same setup for multi-class classification with SVM as in Section 4.1: choose hyper-parameters through cross validation, choose $1/t$ from $\{1, q_{10}, q_{20}, q_{50}\}$, use one-vs-one strategy with Libsvm for classification, and choose SVM regularization from $\left\{10^{-2:1:2}\right\}$. We ran experiments with Intel(R) Xeon(R) CPU E7-8891v3 (2.80GHz) with 256GB RAM.

Results and discussion. The results of SVM for word embedding based document classification for TWITTER, RECIPE, CLASSIC and AMAZON are summarized in Table 6. The performances of k_{TW} is consistently comparative with those of k_{OT}, and better than those of k_{SW}. Moreover, the computational time of k_{TW} is much less than that of k_{OT}. Especially, in CLASSIC dataset, the computation of k_{TW} is less than 3 hours while that of k_{OT} is more than 8 days. Averaging over n_s slices (i.e. n_s randomly tree metrics) for tree-slices-Wasserstein is important since $n_s = 1$ for k_{TW} does not work well for all those datasets. The dimension of the *local* word embedding space is 300. So, it is very loose to approximate $\mathbb{R}^{300}$ with a line as in 1D-sliced-Wasserstein for k_{SW}, which may cause k_{SW} to perform worse on those datasets. We also illustrate a trade-off of performances and computational time for $(n_s, H_\mathcal{T}, \kappa)$ in tree-sliced-Wasserstein distances for k_{TW} and n_s slices for k_{SW} in Table 5 for TWITTER dataset. Further results with different parameters $(n_s, H_\mathcal{T}, \kappa)$ for k_{TW} and n_s slices for k_{SW} on all 4 datasets are placed in the appendix.

Table 6. Results of SVM for word embedding-based document classification on TWITTER, RECIPE, CLASSIC and AMAZON datasets. For each dataset, in the parenthesis the three numbers are corresponding to the number of classes, the number of documents and the maximum number of unique words for each document respectively. For each kernel, the averaged accuracy (%) and standard deviation are shown on its first line while time consumption (seconds) is shown in a parenthesis on its second line. Results are reported for k_{SW} with $n_s = 20$ slices, and for k_{TW} with $n_s = 10$ slices, $H_\mathcal{T} = 6$, and $\kappa = 4$ (parameter of the farthest clustering). We highlight our approach and its results in bold.

	TWITTER (3/3108/26)	RECIPE (15/4370/340)	CLASSIC (4/7093/197)	AMAZON (4/8000/884)
k_{OT}	72.72 ± 0.57 (28522)	52.05 ± 0.61 (305480)	96.59 ± 0.32 (705780)	94.43 ± 0.38 (936320)
k_{SW}	70.10 ± 0.75 (2746)	46.07 ± 0.81 (28099)	92.17 ± 0.42 (33491)	86.28 ± 0.65 (42456)
$\boldsymbol{k_{\text{TW}}}$	**73.09 ± 0.37 (1914)**	**52.25 ± 1.00 (4765)**	**96.85 ± 0.27 (9423)**	**94.24 ± 0.55 (28533)**

5. Conclusion

In this work, we proposed tree-sliced-Wasserstein distances which is an average of randomly ground tree metrics for optimal transport (OT). We showed that OT with tree metrics, can be not only computed in linear time, but is also negative definite, which supports to use kernel methods on OT geometry. Moreover, the proposed tree-Wasserstein kernel compares favorably with other baseline kernels on many benchmark datasets in topological data analysis and word embedding-based document classification.

ACKNOWLEDGMENTS

TL acknowledges the support of JSPS KAKENHI Grant number 17K12745.

References

Adams, H., Emerson, T., Kirby, M., Neville, R., Peterson, C., Shipman, P., Chepushtanova, S., Hanson, E., Motta, F., and Ziegelmeier, L. Persistence images: A stable vector representation of persistent homology. *The Journal of Machine Learning Research*, 18(1):218–252, 2017.

Adler, J. and Lunz, S. Banach wasserstein gan. In *Advances in Neural Information Processing Systems*, pp. 6755–6764. 2018.

Altschuler, J., 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.

Altschuler, J., Bach, F., Rudi, A., and Weed, J. Approximating the quadratic transportation metric in near-linear time. *arXiv preprint arXiv:1810.10046*, 2018a.

Altschuler, J., Bach, F., Rudi, A., and Weed, J. Massively scalable sinkhorn distances via the nystrom method. *arXiv preprint arXiv:1812.05189*, 2018b.

Ambrogioni, L., Guclu, U., Gucluturk, Y., Hinne, M., van Gerven, M. A. J., and Maris, E. Wasserstein variational inference. In *Advances in Neural Information Processing Systems*, pp. 2478–2487. 2018.

Anonymous. What is random packing? *Nature*, 239:488–489, 1972.

Arjovsky, M., Chintala, S., and Bottou, L. Wasserstein generative adversarial networks. In *International Conference on Machine Learning*, pp. 214–223, 2017.

Berg, C., Christensen, J. P. R., and Ressel, P. *Harmonic analysis on semigroups*. Springer-Verlag, 1984.

Bonneel, N., Peyre, G., and Cuturi, M. Wasserstein Barycentric Coordinates: Histogram Regression Using Optimal Transport. *ACM Transactions on Graphics (SIGGRAPH 2016)*, 35(4), 2016.

Burkard, R. E. and Cela, E. Linear assignment problems and extensions. In *Handbook of combinatorial optimization*, pp. 75–149. Springer, 1999.

Carriere, M., Cuturi, M., and Oudot, S. Sliced Wasserstein kernel for persistence diagrams. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 664–673, 2017.

Chang, C.-C. and Lin, C.-J. Libsvm: a library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)*, 2(3):27, 2011.

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

Cuturi, M. and Doucet, A. Fast computation of wasserstein barycenters. In *International Conference on Machine Learning*, pp. 685–693, 2014.

Do Ba, K., Nguyen, H. L., Nguyen, H. N., and Rubinfeld, R. Sublinear time algorithms for earth mover?s distance. *Theory of Computing Systems*, 48(2):428–442, 2011.

Dvurechensky, P., Gasnikov, A., and Kroshnin, A. Computational optimal transport: Complexity by accelerated gradient descent is better than by Sinkhorn's algorithm. In *Proceedings of the 35th International Conference on Machine Learning*, pp. 1367–1376, 2018.

Ebert, J., Spokoiny, V., and Suvorikova, A. Construction of non-asymptotic confidence sets in 2-wasserstein space. *arXiv preprint arXiv:1703.03658*, 2017.

Edelsbrunner, H. and Harer, J. Persistent homology-a survey. *Contemporary mathematics*, 453:257–282, 2008.

Edelsbrunner, H., Letscher, D., and Zomorodian, A. Topological persistence and simplification. In *Proceedings 41st Annual Symposium on Foundations of Computer Science*, pp. 454–463, 2000.

Elliott, S. R. Physics of amorphous materials. *Longman Group, Longman House, Burnt Mill, Harlow, Essex CM 20 2 JE, England, 1983.*, 1983.

Evans, S. N. and Matsen, F. A. The phylogenetic kantorovich–rubinstein metric for environmental sequence samples. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 74(3):569–592, 2012.

Feder, T. and Greene, D. Optimal algorithms for approximate clustering. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, pp. 434–444. ACM, 1988.

Francois, N., Saadatfar, M., Cruikshank, R., and Sheppard, A. Geometrical frustration in amorphous and partially crystallized packings of spheres. *Physical review letters*, 111(14):148001, 2013.

Franklin, J. and Lorenz, J. On the scaling of multidimensional matrices. *Linear Algebra and its applications*, 114: 717–735, 1989.

Frogner, C., Zhang, C., Mobahi, H., Araya, M., and Poggio, T. A. Learning with a wasserstein loss. In *Advances in Neural Information Processing Systems*, pp. 2053–2061, 2015.

Gao, R., Xie, L., Xie, Y., and Xu, H. Robust hypothesis testing using wasserstein uncertainty sets. In *Advances in Neural Information Processing Systems*, pp. 7913–7923. 2018.

Genevay, A., Cuturi, M., Peyre, G., and Bach, F. Stochastic optimization for large-scale optimal transport. In *Advances in Neural Information Processing Systems*, pp. 3440–3448, 2016.

Gonzalez, T. F. Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, 38: 293–306, 1985.

Harchaoui, Z., Moulines, E., and Bach, F. R. Kernel change-point analysis. In *Advances in neural information processing systems*, pp. 609–616, 2009.

Hertzsch, J.-M., Sturman, R., and Wiggins, S. Dna microarrays: design principles for maximizing ergodic, chaotic mixing. *Small*, 3(2):202–218, 2007.

Ho, N., Nguyen, X., Yurochkin, M., Bui, H. H., Huynh, V., and Phung, D. Multilevel clustering via Wasserstein means. In *Proceedings of the 34th International Conference on Machine Learning*, pp. 1501–1509, 2017.

Indyk, P. and Thaper, N. Fast image retrieval via embeddings. *International Workshop on Statistical and Computational Theories of Vision*, 2003.

Johnson, S. C. Hierarchical clustering schemes. *Psychometrika*, 32(3):241–254, 1967.

Kalantari, B., Lari, I., Ricca, F., and Simeone, B. On the complexity of general matrix scaling and entropy minimization via the ras algorithm. *Mathematical Programming*, 112(2):371–401, 2008.

Kantorovich, L. V. On the transfer of masses. In *Dokl. Akad. Nauk. SSSR*, volume 37, pp. 227–229, 1942.

Kloeckner, B. R. A geometric study of wasserstein spaces: ultrametrics. *Mathematika*, 61(1):162–178, 2015.

Kriege, N. M., Giscard, P.-L., and Wilson, R. On valid optimal assignment kernels and applications to graph classification. In *Advances in Neural Information Processing Systems*, pp. 1623–1631, 2016.

Kusano, G., Hiraoka, Y., and Fukumizu, K. Persistence weighted gaussian kernel for topological data analysis. In *International Conference on Machine Learning*, pp. 2004–2013, 2016.

Kusano, G., Fukumizu, K., and Hiraoka, Y. Kernel method for persistence diagrams via kernel embedding and weight factor. *Journal of Machine Learning Research*, 18(189):1–41, 2018.

Kusner, M., Sun, Y., Kolkin, N., and Weinberger, K. From word embeddings to document distances. In *International Conference on Machine Learning*, pp. 957–966, 2015.

Latecki, L. J., Lakamper, R., and Eckhardt, T. Shape descriptors for non-rigid shapes with a single closed contour. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pp. 424–429, 2000.

Lavenant, H., Claici, S., Chien, E., and Solomon, J. Dynamical optimal transport on discrete surfaces. In *SIGGRAPH Asia 2018 Technical Papers*, pp. 250. ACM, 2018.

Le, T. and Yamada, M. Persistence Fisher kernel: A Riemannian manifold kernel for persistence diagrams. In *Advances in Neural Information Processing Systems*, pp. 10028–10039, 2018.

Lee, J. and Raginsky, M. Minimax statistical learning with wasserstein distances. In *Advances in Neural Information Processing Systems*, pp. 2692–2701. 2018.

Linial, N., Samorodnitsky, A., and Wigderson, A. A deterministic strongly polynomial algorithm for matrix scaling and approximate permanents. *Combinatorica*, 20(4):545–568, 2000.

Lozupone, C. and Knight, R. Unifrac: a new phylogenetic method for comparing microbial communities. *Applied and environmental microbiology*, 71(12):8228–8235, 2005.

Lozupone, C. A., Hamady, M., Kelley, S. T., and Knight, R. Quantitative and qualitative β diversity measures lead to different insights into factors that structure microbial communities. *Applied and environmental microbiology*, 73(5):1576–1585, 2007.

McGregor, A. and Stubbs, D. Sketching earth-mover distance on graph metrics. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pp. 274–286. Springer, 2013.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pp. 3111–3119, 2013.

Morariu, V. I., Srinivasan, B. V., Raykar, V. C., Duraiswami, R., and Davis, L. S. Automatic online tuning for fast gaussian summation. In *Advances in neural information processing systems*, pp. 1113–1120, 2009.

Nakamura, T., Hiraoka, Y., Hirata, A., Escolar, E. G., and Nishiura, Y. Persistent homology and many-body atomic structure for medium-range order in the glass. *Nanotechnology*, 26(30):304001, 2015.

Panaretos, V. M., Zemel, Y., et al. Amplitude and phase variation of point processes. *The Annals of Statistics*, 44 (2):771–812, 2016.

Pele, O. and Werman, M. Fast and robust earth mover's distances. In *International Conference on Computer Vision*, pp. 460–467, 2009.

Peyré, G. and Cuturi, M. Computational optimal transport. *Foundations and Trends in Machine Learning*, 11(5-6), 2019.

Reininghaus, J., Huber, S., Bauer, U., and Kwitt, R. A stable multi-scale kernel for topological machine learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pp. 4741–4748, 2015.

Rubner, Y., Tomasi, C., and Guibas, L. J. The earth mover's distance as a metric for image retrieval. *International journal of computer vision*, 40(2):99–121, 2000.

Salton, G. and Buckley, C. Term-weighting approaches in automatic text retrieval. *Information processing & management*, 24(5):513–523, 1988.

Santambrogio, F. *Optimal transport for applied mathematicians*. Birkhauser, 2015.

Semple, C. and Steel, M. Phylogenetics. *Oxford Lecture Series in Mathematics and its Applications*, 2003.

Shkarin, S. A. Isometric embedding of finite ultrametric spaces in banach spaces. *Topology and its Applications*, 142(1-3):13–17, 2004.

Solomon, J., Rustamov, R., Guibas, L., and Butscher, A. Earth mover's distances on discrete surfaces. *ACM Transactions on Graphics (TOG)*, 33(4):67, 2014a.

Solomon, J., Rustamov, R., Guibas, L., and Butscher, A. Wasserstein propagation for semi-supervised learning. In *International Conference on Machine Learning*, pp. 306–314, 2014b.

Solomon, J., De Goes, F., Peyre, G., Cuturi, M., Butscher, A., Nguyen, A., Du, T., and Guibas, L. Convolutional wasserstein distances: Efficient optimal transportation on geometric domains. *ACM Transactions on Graphics (TOG)*, 34(4):66, 2015.

Sommerfeld, M. and Munk, A. Inference for empirical wasserstein distances on finite spaces. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 80(1):219–238, 2018.

Tarjan, R. E. Dynamic trees as search trees via euler tours, applied to the network simplex algorithm. *Mathematical Programming*, 78(2):169–177, 1997.

Tenetov, E., Wolansky, G., and Kimmel, R. Fast entropic regularized optimal transport using semidiscrete cost approximation. *SIAM Journal on Scientific Computing*, 40 (5):A3400–A3422, 2018.

Turner, K., Mukherjee, S., and Boyer, D. M. Persistent homology transform for modeling shapes and surfaces. *Information and Inference: A Journal of the IMA*, 3(4): 310–344, 2014.

Villani, C. *Topics in optimal transportation*. Number 58. American Mathematical Soc., 2003.

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

Yang, C., Duraiswami, R., and Davis, L. S. Efficient kernel machines using the improved fast gauss transform. In *Advances in neural information processing systems*, pp. 1561–1568, 2005.

Zhang, J., Marszalek, M., Lazebnik, S., and Schmid, C. Local features and kernels for classification of texture and object categories: A comprehensive study. *International journal of computer vision*, 73(2):213–238, 2007.

Table 7. Computational time (seconds) for different parameters $(n_s, H_\mathcal{T})$ for tree-sliced-Wasserstein distances in k_{TW} on granular packing system dataset.

	$n_s = 1$	$n_s = 6$	$n_s = 12$	$n_s = 18$	$n_s = 24$	$n_s = 30$
$H_\mathcal{T} = 3$	0.06	0.33	0.75	1.07	1.59	1.99
$H_\mathcal{T} = 4$	0.08	0.47	0.95	1.52	2.16	2.98
$H_\mathcal{T} = 5$	0.12	0.76	1.40	2.13	2.89	3.83
$H_\mathcal{T} = 6$	0.17	1.09	2.01	3.03	3.85	4.01

A. Optimal Transport for Persistence Diagrams

Let $\text{Dg}_i = (x_1, x_2, \ldots, x_n)$ and $\text{Dg}_j = (z_1, z_2, \ldots, z_m)$ be two persistence diagrams where $x_i, z_j \in \mathbb{R}^2$, and $\Theta = \{(a, a) \mid a \in \mathbb{R}\}$ be the diagonal set. Denote $\text{Dg}_{i\Theta} = \{\Pi_\Theta(x) \mid x \in \text{Dg}_i\}$ where $\Pi_\Theta(x)$ is a projection of x on Θ.

In the bottle neck distance between two persistence diagrams Dg_i and Dg_j, the transportation plan is bijective between $(\text{Dg}_i \cup \Theta)$ and $(\text{Dg}_j \cup \Theta)$ instead of between Dg_i and Dg_j since two persistence diagrams may have different masses. Additionally, (Carriere et al., 2017) use a transportation plan between $(\text{Dg}_i \cup \text{Dg}_{j\Theta})$ and $(\text{Dg}_j \cup \text{Dg}_{i\Theta})$ for sliced-Wasserstein distance between Dg_i and Dg_j. In our implementation for the tree-Wasserstein distance and optimal transport distance between Dg_i and Dg_j, we followed (Carriere et al., 2017) to use a transportation plan between $(\text{Dg}_i \cup \text{Dg}_{j\Theta})$ and $(\text{Dg}_j \cup \text{Dg}_{i\Theta})$ for those distances[8].

B. Negative Definiteness of L_1 Distance

For two real numbers u, v, the function $(u, v) \mapsto (u - v)^2$ is obviously negative definite. Following (Berg et al., 1984) (Corollary 2.10, p.78), the function $(u, v) \mapsto |u - v|$ is negative definite. Therefore, L_1 distance is a sum of negative definite functions. Thus, L_1 distance is negative definite.

C. More Experimental Results

We provide many more experimental results for our proposed tree-Wasserstein kernel on topological data analysis (TDA) and word embedding-based document classification.

C.1. Change Point Detection for Material Data Analysis (TDA)

- Table 7 shows computational time for different parameters $(n_s, H_\mathcal{T})$ for tree-sliced-Wasserstein distances in k_{TW} on granular packing system dataset.
- Table 8 shows computational time for different parameters $(n_s, H_\mathcal{T})$ for tree-sliced-Wasserstein distances in k_{TW} on SiO_2 dataset.

C.2. Word Embedding-based Document Classification

- Table 9 shows SVM results and computational time for different parameters $(n_s, H_\mathcal{T}, \kappa)$ for tree-sliced-Wasserstein distances in k_{TW}, and different n_s for k_{SW} on `TWITTER` dataset.
- Table 10 shows SVM results and computational time for different parameters $(n_s, H_\mathcal{T}, \kappa)$ for tree-sliced-Wasserstein distances in k_{TW}, and different n_s for k_{SW} on `RECIPE` dataset.
- Table 11 shows SVM results and computational time for different parameters $(n_s, H_\mathcal{T}, \kappa)$ for tree-sliced-Wasserstein distances in k_{TW}, and different n_s for k_{SW} on `CLASSIC` dataset.
- Table 12 shows SVM results and computational time for different parameters $(n_s, H_\mathcal{T}, \kappa)$ for tree-sliced-Wasserstein distances in k_{TW}, and different n_s for k_{SW} on `AMAZON` dataset.

[8] Le & Yamada (2018) also followed this line of work to define Fisher information metric between two persistence diagrams.

Table 8. Computational time (seconds) for different parameters $(n_s, H_\mathcal{T})$ for tree-sliced-Wasserstein distances in k_{TW} on SiO_2 dataset.

	$n_s = 1$	$n_s = 6$	$n_s = 12$	$n_s = 18$	$n_s = 24$	$n_s = 30$
$H_\mathcal{T} = 3$	0.21	1.24	2.64	3.67	4.93	6.09
$H_\mathcal{T} = 4$	0.27	1.58	3.46	5.03	7.05	7.91
$H_\mathcal{T} = 5$	0.37	2.20	4.50	6.53	8.59	11.64
$H_\mathcal{T} = 6$	0.53	3.25	6.32	10.06	13.71	15.28

Table 9. Results of SVM with different parameters $(n_s, H_\mathcal{T}, \kappa)$ for k_{TW} and different n_s for k_{TW} on `TWITTER` dataset. The averaged accuracy (%) and standard deviation are shown on its first line while time consumption (seconds) is shown in a parenthesis on its second line.

	$n_s = 1$	$n_s = 5$	$n_s = 10$	$n_s = 20$	$n_s = 30$	$n_s = 50$
k_{TW} $(H_\mathcal{T} = 8, \kappa = 2)$	69.73 ± 0.29 (10)	71.65 ± 0.42 (46)	71.42 ± 0.48 (98)	72.19 ± 0.56 (206)	72.47 ± 1.70 (316)	72.95 ± 0.49 (473)
k_{TW} $(H_\mathcal{T} = 9, \kappa = 2)$	70.00 ± 0.39 (21)	71.66 ± 0.61 (105)	72.43 ± 0.72 (208)	72.96 ± 0.50 (419)	72.82 ± 0.58 (624)	73.13 ± 0.49 (1066)
k_{TW} $(H_\mathcal{T} = 10, \kappa = 2)$	70.61 ± 0.52 (49)	72.23 ± 0.52 (255)	72.79 ± 0.58 (516)	72.95 ± 1.01 (1173)	72.86 ± 0.45 (1555)	72.95 ± 0.44 (2451)
k_{TW} $(H_\mathcal{T} = 11, \kappa = 2)$	71.24 ± 0.55 (129)	72.42 ± 0.62 (537)	72.91 ± 0.72 (1105)	72.85 ± 0.91 (2213)	72.86 ± 0.33 (3511)	73.37 ± 0.60 (6013)
k_{TW} $(H_\mathcal{T} = 12, \kappa = 2)$	71.21 ± 0.49 (324)	72.44 ± 0.61 (1613)	72.88 ± 0.63 (3299)	73.27 ± 0.71 (6469)	73.21 ± 0.58 (9862)	73.46 ± 0.67 (16542)
k_{TW} $(H_\mathcal{T} = 5, \kappa = 3)$	70.75 ± 0.58 (13)	71.97 ± 0.66 (52)	71.68 ± 0.51 (118)	72.46 ± 0.50 (256)	72.70 ± 0.57 (362)	72.71 ± 0.54 (434)
k_{TW} $(H_\mathcal{T} = 6, \kappa = 3)$	70.49 ± 0.48 (33)	72.15 ± 0.48 (169)	72.47 ± 0.58 (335)	72.90 ± 0.55 (674)	73.35 ± 0.35 (1016)	73.15 ± 0.58 (1687)
k_{TW} $(H_\mathcal{T} = 7, \kappa = 3)$	71.68 ± 0.65 (133)	72.45 ± 0.57 (633)	73.08 ± 0.55 (1294)	73.62 ± 0.43 (2509)	73.25 ± 0.34 (3785)	73.50 ± 0.69 (6134)
k_{TW} $(H_\mathcal{T} = 8, \kappa = 3)$	72.01 ± 0.55 (437)	72.89 ± 0.80 (2188)	72.35 ± 0.57 (4364)	73.21 ± 0.60 (8896)	73.33 ± 0.71 (13277)	73.33 ± 0.52 (21787)
k_{TW} $(H_\mathcal{T} = 4, \kappa = 4)$	69.74 ± 0.41 (12)	71.68 ± 0.26 (56)	71.66 ± 0.50 (113)	72.28 ± 0.76 (239)	72.94 ± 0.80 (364)	72.79 ± 0.62 (605)
k_{TW} $(H_\mathcal{T} = 5, \kappa = 4)$	71.57 ± 0.71 (46)	72.67 ± 0.42 (231)	72.86 ± 0.78 (472)	73.24 ± 0.43 (939)	73.37 ± 0.35 (1469)	73.42 ± 0.42 (2400)
k_{TW} $(H_\mathcal{T} = 6, \kappa = 4)$	72.81 ± 0.65 (180)	72.82 ± 0.51 (929)	73.09 ± 0.37 (1914)	73.67 ± 0.63 (3703)	73.37 ± 0.39 (5569)	73.54 ± 0.81 (9166)
k_{TW} $(H_\mathcal{T} = 3, \kappa = 5)$	69.09 ± 0.14 (5)	71.40 ± 0.48 (21)	72.16 ± 0.52 (43)	72.19 ± 0.64 (80)	72.50 ± 0.65 (116)	72.99 ± 0.49 (203)
k_{TW} $(H_\mathcal{T} = 4, \kappa = 5)$	71.19 ± 0.52 (29)	72.26 ± 0.68 (155)	72.42 ± 0.61 (307)	73.14 ± 0.72 (608)	73.34 ± 0.41 (927)	73.30 ± 0.38 (1527)
k_{TW} $(H_\mathcal{T} = 5, \kappa = 5)$	71.87 ± 0.57 (187)	72.82 ± 0.51 (966)	73.13 ± 0.33 (1785)	73.43 ± 0.55 (3248)	73.31 ± 0.38 (4842)	73.74 ± 0.70 (8089)
k_{SW}	68.86 ± 0.13 (152)	68.85 ± 0.11 (669)	68.89 ± 0.14 (1316)	70.10 ± 0.75 (2746)	69.61 ± 0.47 (4097)	70.04 ± 0.40 (6635)

Table 10. Results of SVM with different parameters $(n_s, H_\mathcal{T}, \kappa)$ for k_{TW} and different n_s for k_{TW} on RECIPE dataset. The averaged accuracy (%) and standard deviation are shown on its first line while time consumption (seconds) is shown in a parenthesis on its second line.

	$n_s = 1$	$n_s = 5$	$n_s = 10$	$n_s = 20$	$n_s = 30$	$n_s = 50$
k_{TW} $(H_\mathcal{T} = 8, \kappa = 2)$	43.98 ± 0.66 (34)	49.03 ± 0.70 (178)	50.23 ± 0.45 (327)	50.63 ± 0.81 (388)	50.90 ± 0.60 (963)	51.43 ± 0.67 (1662)
k_{TW} $(H_\mathcal{T} = 9, \kappa = 2)$	45.12 ± 0.45 (63)	50.27 ± 0.52 (339)	50.96 ± 0.81 (624)	51.13 ± 0.71 (1254)	51.80 ± 0.84 (1991)	52.58 ± 0.80 (3273)
k_{TW} $(H_\mathcal{T} = 10, \kappa = 2)$	44.48 ± 0.50 (136)	49.07 ± 1.00 (696)	50.37 ± 0.42 (1431)	51.98 ± 1.05 (2878)	51.81 ± 0.62 (4230)	52.44 ± 0.35 (6957)
k_{TW} $(H_\mathcal{T} = 11, \kappa = 2)$	46.19 ± 0.73 (320)	51.07 ± 0.39 (1554)	51.98 ± 0.80 (3065)	51.50 ± 0.37 (5905)	52.45 ± 0.40 (8843)	52.14 ± 0.59 (14578)
k_{TW} $(H_\mathcal{T} = 12, \kappa = 2)$	48.70 ± 0.70 (719)	51.63 ± 0.56 (3569)	51.34 ± 0.64 (7193)	52.08 ± 0.70 (14091)	52.46 ± 0.61 (21462)	52.79 ± 0.49 (35654)
k_{TW} $(H_\mathcal{T} = 5, \kappa = 3)$	42.94 ± 0.94 (38)	47.97 ± 0.51 (218)	50.82 ± 0.64 (436)	50.70 ± 0.86 (733)	51.78 ± 0.75 (1363)	52.25 ± 0.62 (2050)
k_{TW} $(H_\mathcal{T} = 6, \kappa = 3)$	44.63 ± 0.90 (108)	51.00 ± 0.47 (503)	52.10 ± 0.64 (981)	51.91 ± 0.88 (1663)	51.95 ± 0.58 (2895)	52.05 ± 0.68 (5221)
k_{TW} $(H_\mathcal{T} = 7, \kappa = 3)$	46.45 ± 0.94 (281)	49.80 ± 0.67 (1530)	51.92 ± 0.87 (2952)	51.92 ± 0.60 (5922)	52.71 ± 0.81 (8764)	52.78 ± 0.83 (13835)
k_{TW} $(H_\mathcal{T} = 8, \kappa = 3)$	48.21 ± 0.90 (694)	52.32 ± 0.73 (3685)	52.88 ± 0.55 (7289)	53.19 ± 1.03 (14552)	52.53 ± 0.68 (21797)	53.20 ± 0.43 (37071)
k_{TW} $(H_\mathcal{T} = 4, \kappa = 4)$	41.88 ± 0.87 (25)	50.57 ± 0.73 (126)	51.31 ± 0.64 (249)	51.59 ± 0.79 (517)	51.54 ± 0.66 (764)	52.37 ± 0.63 (1252)
k_{TW} $(H_\mathcal{T} = 5, \kappa = 4)$	46.54 ± 0.87 (109)	51.54 ± 0.63 (551)	52.04 ± 0.56 (1100)	52.30 ± 0.94 (2204)	52.05 ± 0.65 (3277)	52.11 ± 0.66 (5511)
k_{TW} $(H_\mathcal{T} = 6, \kappa = 4)$	49.47 ± 0.67 (476)	52.32 ± 0.48 (2370)	52.25 ± 1.00 (4765)	52.36 ± 0.70 (9171)	52.91 ± 0.85 (14284)	53.12 ± 0.70 (23847)
k_{TW} $(H_\mathcal{T} = 3, \kappa = 5)$	43.03 ± 0.50 (17)	48.54 ± 0.90 (81)	50.24 ± 0.91 (193)	51.46 ± 0.16 (384)	52.03 ± 0.55 (536)	51.26 ± 0.62 (886)
k_{TW} $(H_\mathcal{T} = 4, \kappa = 5)$	45.61 ± 0.82 (71)	50.72 ± 0.57 (335)	51.98 ± 0.48 (705)	51.97 ± 0.85 (1414)	52.22 ± 0.69 (2123)	52.24 ± 0.63 (3453)
k_{TW} $(H_\mathcal{T} = 5, \kappa = 5)$	48.25 ± 0.88 (291)	52.56 ± 0.61 (1488)	52.55 ± 0.56 (3079)	52.60 ± 0.71 (5966)	53.20 ± 0.80 (8503)	53.40 ± 0.74 (15179)
k_{SW}	31.45 ± 0.53 (1470)	39.42 ± 1.22 (7381)	42.67 ± 0.99 (14028)	46.07 ± 0.81 (28099)	46.38 ± 0.53 (41241)	47.85 ± 0.51 (73889)

Table 11. Results of SVM with different parameters $(n_s, H_\mathcal{T}, \kappa)$ for k_{TW} and different n_s for k_{TW} on CLASSIC dataset. The averaged accuracy (%) and standard deviation are shown on its first line while time consumption (seconds) is shown in a parenthesis on its second line.

	$n_s = 1$	$n_s = 5$	$n_s = 10$	$n_s = 20$	$n_s = 30$	$n_s = 50$
k_{TW} $(H_\mathcal{T} = 8, \kappa = 2)$	90.05 ± 0.35 (115)	95.26 ± 0.36 (481)	96.00 ± 0.32 (1016)	96.42 ± 0.36 (2046)	96.75 ± 0.30 (2882)	96.67 ± 0.29 (4893)
k_{TW} $(H_\mathcal{T} = 9, \kappa = 2)$	92.27 ± 0.34 (169)	95.91 ± 0.57 (882)	96.38 ± 0.22 (1755)	96.59 ± 0.24 (3454)	96.53 ± 0.31 (5157)	96.67 ± 0.28 (8594)
k_{TW} $(H_\mathcal{T} = 10, \kappa = 2)$	93.84 ± 0.52 (371)	96.16 ± 0.48 (1859)	96.26 ± 0.39 (3886)	96.79 ± 0.24 (7944)	96.71 ± 0.43 (11916)	96.81 ± 0.20 (19456)
k_{TW} $(H_\mathcal{T} = 11, \kappa = 2)$	94.44 ± 0.35 (818)	95.15 ± 0.30 (4033)	96.33 ± 0.42 (8483)	96.77 ± 0.33 (18037)	96.53 ± 0.32 (27099)	96.77 ± 0.30 (45963)
k_{TW} $(H_\mathcal{T} = 12, \kappa = 2)$	94.61 ± 0.52 (2919)	96.42 ± 0.35 (14963)	96.63 ± 0.29 (30308)	96.68 ± 0.38 (62199)	96.85 ± 0.38 (93383)	96.65 ± 0.31 (156010)
k_{TW} $(H_\mathcal{T} = 5, \kappa = 3)$	93.06 ± 0.32 (73)	96.06 ± 0.39 (389)	96.27 ± 0.39 (787)	96.22 ± 0.30 (1568)	96.54 ± 0.33 (2420)	96.79 ± 0.30 (3859)
k_{TW} $(H_\mathcal{T} = 6, \kappa = 3)$	94.12 ± 0.49 (223)	96.12 ± 0.30 (949)	96.49 ± 0.43 (1678)	96.83 ± 0.24 (3494)	96.71 ± 0.31 (4960)	96.76 ± 0.37 (8189)
k_{TW} $(H_\mathcal{T} = 7, \kappa = 3)$	94.72 ± 0.43 (484)	96.35 ± 0.45 (2378)	96.90 ± 0.36 (4744)	96.79 ± 0.23 (9523)	96.68 ± 0.28 (14250)	96.81 ± 0.25 (23751)
k_{TW} $(H_\mathcal{T} = 8, \kappa = 3)$	95.25 ± 0.51 (1364)	96.64 ± 0.45 (6973)	96.63 ± 0.30 (13235)	96.88 ± 0.24 (26449)	96.71 ± 0.43 (42520)	96.84 ± 0.25 (75548)
k_{TW} $(H_\mathcal{T} = 4, \kappa = 4)$	92.47 ± 0.54 (85)	95.62 ± 0.37 (397)	96.36 ± 0.33 (818)	96.46 ± 0.20 (1502)	96.77 ± 0.34 (2333)	96.84 ± 0.35 (3543)
k_{TW} $(H_\mathcal{T} = 5, \kappa = 4)$	93.29 ± 0.57 (273)	96.46 ± 0.32 (1363)	96.83 ± 0.38 (2631)	96.82 ± 0.23 (5399)	96.99 ± 0.30 (7970)	96.92 ± 0.36 (13320)
k_{TW} $(H_\mathcal{T} = 6, \kappa = 4)$	95.69 ± 0.31 (945)	96.49 ± 0.42 (4715)	96.85 ± 0.27 (9423)	96.91 ± 0.33 (18163)	96.74 ± 0.37 (26560)	96.84 ± 0.20 (43352)
k_{TW} $(H_\mathcal{T} = 3, \kappa = 5)$	92.47 ± 0.54 (29)	95.62 ± 0.37 (115)	96.36 ± 0.33 (236)	96.46 ± 0.20 (459)	96.77 ± 0.34 (740)	96.84 ± 0.35 (1183)
k_{TW} $(H_\mathcal{T} = 4, \kappa = 5)$	93.29 ± 0.57 (138)	96.46 ± 0.32 (744)	96.83 ± 0.38 (1456)	96.82 ± 0.23 (2744)	96.99 ± 0.30 (4391)	96.92 ± 0.36 (7370)
k_{TW} $(H_\mathcal{T} = 5, \kappa = 5)$	95.69 ± 0.39 (749)	96.49 ± 0.42 (3767)	96.85 ± 0.27 (7456)	96.91 ± 0.33 (14746)	96.74 ± 0.37 (22735)	96.84 ± 0.20 (37339)
k_{SW}	76.58 ± 0.29 (1715)	86.19 ± 0.46 (8347)	89.51 ± 0.50 (16311)	92.17 ± 0.42 (33491)	92.74 ± 0.46 (47855)	94.13 ± 0.35 (79558)

Table 12. Results of SVM with different parameters $(n_s, H_\mathcal{T}, \kappa)$ for k_{TW} and different n_s for k_{TW} on AMAZON dataset. The averaged accuracy (%) and standard deviation are shown on its first line while time consumption (seconds) is shown in a parenthesis on its second line.

	$n_s = 1$	$n_s = 5$	$n_s = 10$	$n_s = 20$	$n_s = 30$	$n_s = 50$
k_{TW} $(H_\mathcal{T} = 8, \kappa = 2)$	81.91 ± 0.61 (201)	91.40 ± 0.56 (1001)	92.18 ± 0.29 (1927)	93.17 ± 0.40 (3050)	94.25 ± 0.40 (5733)	94.38 ± 0.31 (9308)
k_{TW} $(H_\mathcal{T} = 9, \kappa = 2)$	85.28 ± 0.61 (314)	91.52 ± 0.56 (1632)	93.28 ± 0.46 (3161)	94.04 ± 0.36 (6453)	94.24 ± 0.66 (96.28)	94.14 ± 0.26 (16355)
k_{TW} $(H_\mathcal{T} = 10, \kappa = 2)$	87.10 ± 0.41 (799)	92.90 ± 0.70 (4048)	93.49 ± 0.43 (8133)	94.02 ± 0.50 (16038)	94.54 ± 0.58 (24408)	94.32 ± 0.23 (41278)
k_{TW} $(H_\mathcal{T} = 11, \kappa = 2)$	87.97 ± 0.58 (2023)	93.04 ± 0.49 (9514)	93.63 ± 0.39 (18649)	94.24 ± 0.45 (37102)	94.29 ± 0.41 (56200)	94.30 ± 0.44 (94515)
k_{TW} $(H_\mathcal{T} = 12, \kappa = 2)$	89.48 ± 0.75 (4192)	93.24 ± 0.44 (20792)	93.68 ± 0.48 (40610)	94.19 ± 0.44 (80897)	94.35 ± 0.43 (122150)	94.61 ± 0.36 (206360)
k_{TW} $(H_\mathcal{T} = 5, \kappa = 3)$	83.17 ± 0.76 (316)	91.78 ± 0.48 (1573)	93.00 ± 0.37 (3011)	93.83 ± 0.39 (6208)	94.09 ± 0.53 (9580)	94.34 ± 0.50 (15060)
k_{TW} $(H_\mathcal{T} = 6, \kappa = 3)$	86.90 ± 0.46 (464)	92.46 ± 0.58 (2156)	93.63 ± 0.45 (4585)	94.20 ± 0.42 (8862)	94.20 ± 0.37 (12750)	94.34 ± 0.33 (20935)
k_{TW} $(H_\mathcal{T} = 7, \kappa = 3)$	89.56 ± 0.79 (1642)	93.06 ± 0.62 (8042)	93.97 ± 0.49 (15883)	94.37 ± 0.35 (31787)	94.46 ± 0.69 (47578)	94.20 ± 0.20 (81293)
k_{TW} $(H_\mathcal{T} = 8, \kappa = 3)$	90.12 ± 0.51 (5207)	93.67 ± 0.30 (25386)	94.01 ± 0.50 (51558)	94.40 ± 0.48 (101710)	94.58 ± 0.59 (147800)	94.35 ± 0.28 (244380)
k_{TW} $(H_\mathcal{T} = 4, \kappa = 4)$	84.80 ± 0.71 (226)	91.78 ± 0.37 (1022)	93.02 ± 0.48 (2025)	94.02 ± 0.47 (4192)	93.98 ± 0.38 (6473)	94.31 ± 0.65 (10548)
k_{TW} $(H_\mathcal{T} = 5, \kappa = 4)$	87.90 ± 0.45 (605)	92.90 ± 0.57 (2954)	93.99 ± 0.35 (5786)	94.09 ± 0.53 (11899)	94.67 ± 0.39 (18208)	94.39 ± 0.32 (26594)
k_{TW} $(H_\mathcal{T} = 6, \kappa = 4)$	89.92 ± 0.54 (2903)	93.57 ± 0.51 (14958)	94.24 ± 0.55 (28533)	94.34 ± 0.42 (57910)	94.67 ± 0.69 (89811)	94.48 ± 0.29 (145380)
k_{TW} $(H_\mathcal{T} = 3, \kappa = 5)$	84.64 ± 0.39 (44)	91.01 ± 0.38 (208)	92.83 ± 0.30 (427)	93.92 ± 0.27 (930)	94.12 ± 0.48 (1403)	94.41 ± 0.68 (2219)
k_{TW} $(H_\mathcal{T} = 4, \kappa = 5)$	87.11 ± 0.60 (154)	92.73 ± 0.41 (864)	93.66 ± 0.34 (1843)	94.19 ± 0.49 (3632)	94.45 ± 0.42 (5545)	94.35 ± 0.46 (9813)
k_{TW} $(H_\mathcal{T} = 5, \kappa = 5)$	90.83 ± 0.54 (958)	93.32 ± 0.53 (4773)	94.37 ± 0.35 (9651)	94.42 ± 0.31 (18052)	94.68 ± 0.71 (27974)	94.40 ± 0.33 (48468)
k_{SW}	68.29 ± 0.77 (2390)	77.62 ± 1.03 (10825)	82.27 ± 0.69 (21048)	86.28 ± 0.65 (42456)	87.78 ± 0.45 (58084)	89.40 ± 0.48 (103640)