Supervised Tree-Wasserstein Distance

Yuki Takezawa [1 2]   **Ryoma Sato** [1 2]   **Makoto Yamada** [1 2]

Abstract

To measure the similarity of documents, the Wasserstein distance is a powerful tool, but it requires a high computational cost. Recently, for fast computation of the Wasserstein distance, methods for approximating the Wasserstein distance using a tree metric have been proposed. These tree-based methods allow fast comparisons of a large number of documents; however, they are unsupervised and do not learn task-specific distances. In this work, we propose the Supervised Tree-Wasserstein (STW) distance, a fast, supervised metric learning method based on the tree metric. Specifically, we rewrite the Wasserstein distance on the tree metric by the parent–child relationships of a tree, and formulate it as a continuous optimization problem using a contrastive loss. Experimentally, we show that the STW distance can be computed fast, and improves the accuracy of document classification tasks. Furthermore, the STW distance is formulated by matrix multiplications, runs on a GPU, and is suitable for batch processing. Therefore, we show that the STW distance is extremely efficient when comparing a large number of documents.

1. Introduction

The Wasserstein distance is a powerful tool for measuring distances between distributions. It has recently been applied in many fields, such as feature matching (Sarlin et al., 2020; Liu et al., 2020), generative models (Kolouri et al., 2019b), similarity metrics (Kusner et al., 2015; Huang et al., 2016; Yurochkin et al., 2019), and so on.

The Wasserstein distance can be computed by solving the optimal transport problem. For similarity metrics of documents, Kusner et al. (2015) proposed the Word Mover's Distance (WMD). Given the word embedding vectors (Mikolov et al., 2013) and a normalized bag-of-words, the WMD is

the cost of the optimal transport between two documents in the word embedding space. WMD has been used for document classification tasks and has achieved high k-nearest neighbors (kNN) accuracy.

To solve the optimal transport problem, linear programming can be used. However, using linear programming requires cubic time with respect to the number of data points (Pele & Werman, 2009). Cuturi (2013) proposed to add entropic regularization to the optimal transport problem, which can be solved by using a matrix scaling algorithm in quadratic time. To further reduce the computational cost of the optimal transport problem, there are two main strategies. (1) The first approach is to relax the constraint of the optimal transport problem. Specifically, Kusner et al. (2015) relaxed the constraints of the optimal transport problem and transport the mass of each coordinate to the nearest coordinate, called the Relaxed WMD (RWMD). Atasu & Mittelholzer (2019) attached additional constraints to RWMD and proposed a more accurate approximation of WMD. (2) The second approach is to construct a tree metric and compute the Wasserstein distance on the tree metric (tree-Wasserstein distance). Indyk & Thaper (2003) proposed a method to embed the coordinates into the tree metric, called Quadtree. Recently, Le et al. (2019) proposed a method to sample tree metrics and achieved a high accuracy in document classification tasks. Backurs et al. (2020) proposed a more accurate method than Quadtree. These tree-based methods aim to approximate the Wasserstein distance on the Euclidean metric with the tree-Wasserstein distance. The tree-Wasserstein distance can be computed in linear time with respect to the number of nodes in the tree and can quickly compare a large number of documents.

In general, the similarity between documents must be designed in a task-specific manner. However, the methods mentioned above are unsupervised and do not learn task-specific distances. Huang et al. (2016) proposed supervised metric learning based on WMD, called Supervised WMD (S-WMD). S-WMD learns a task-specific distance by leveraging the label information of documents, and improves the kNN accuracy. However, it requires quadratic time to compute S-WMD and there is no supervised metric learning for the tree-Wasserstein distance. Moreover, for the tree-Wasserstein distance, it is challenging to construct the tree metric by leveraging the label information of documents.

---
[1]Kyoto University [2]RIKEN AIP. Correspondence to: Yuki Takezawa <yuki-takezawa@ml.ist.i.kyoto-u.ac.jp>.

In this work, we propose the *Supervised Tree-Wasserstein* (STW) distance, a fast supervised metric learning method for the tree metric. To this end, we propose the *soft tree-Wasserstein distance*, which is a soft variant of the tree-Wasserstein distance. Specifically, we rewrite the tree-Wasserstein distance by the probability of the parent—child relationships of a tree. We then consider learning the probability of the parent–child relationships of a tree by leveraging the label information of documents. By virtue of the soft tree-Wasserstein distance, the STW distance is end-to-end trainable using backpropagation and is formulated only by matrix multiplications, which can be implemented with simple operations on a GPU. Thus, the STW distance is suitable for batch processing and can simultaneously compare multiple documents. Through synthetic and real-world experiments on document classification tasks, we show that the STW distance can build a tree that represents the task-specific distance and has improved accuracy. Furthermore, we show that the STW distance is more efficient than the existing methods for computing Wasserstein distances, especially when comparing a large number of documents.

Our contributions are as follows:

- We propose a soft variant of the tree-Wasserstein distance, which is differentiable with respect to the probability of the parent–child relationships of a tree. It can be computed by simple operations on a GPU and is suitable for batch processing.

- Using the soft variant of the tree-Wasserstein distance, we propose fast supervised metric learning for a tree metric, which is formulated as a continuous optimization problem.

- Experimentally, we show that our method is fast and improves the accuracy of document classification tasks.

Notation: In the following sections, we write $\mathbf{1}_n$ for an n-dimensional vector with all ones, $\mathbf{0}_n$ for an n-dimensional vector with all zeros, $\mathbf{I}$ for the identity matrix, and δ for the Dirac delta function.

2. Related Work

In this section, we introduce the existing Wasserstein distances and the methods for continuous optimization for learning a tree structure, and then present their drawbacks.

2.1. Wasserstein Distances

Given a simplex $\mathbf{a} \in \mathbb{R}_+^n$ and $\mathbf{b} \in \mathbb{R}_+^m$, we write $U(\mathbf{a}, \mathbf{b})$ for the transport polytope of $\mathbf{a}$ and $\mathbf{b}$ as follows:

$$U(\mathbf{a}, \mathbf{b}) = \{\mathbf{T} \in \mathbb{R}_+^{n \times m} \mid \mathbf{T}\mathbf{1}_m = \mathbf{a}, \mathbf{T}^\top \mathbf{1}_n = \mathbf{b}\},$$

Given a cost $c(\mathbf{x}_i, \mathbf{x}_j)$ between coordinates $\mathbf{x}_i$ and $\mathbf{x}_j$, the optimal transport problem between $\mathbf{a}$ and $\mathbf{b}$ is defined as follows:

$$\min_{\mathbf{T} \in U(\mathbf{a}, \mathbf{b})} \sum_{i,j} \mathbf{T}_{i,j} \, c(\mathbf{x}_i, \mathbf{x}_j).$$

If $c(\mathbf{x}_i, \mathbf{x}_j)$ is a metric, then the cost of the optimal transport is a metric, which is a special case of Wasserstein distances.

In document classification tasks, given word embedding vectors $\mathbf{x}_i$ and $\mathbf{x}_j$, Kusner et al. (2015) defined the cost $c(\mathbf{x}_i, \mathbf{x}_j) = \|\mathbf{x}_i - \mathbf{x}_j\|_2^2$ and simplex $\mathbf{a}$ and $\mathbf{b}$ as the normalized bag-of-words, and proposed to use the optimal transport cost as the dissimilarity of documents, called Word Mover's Distance (WMD). To further improve the classification accuracy, Huang et al. (2016) proposed supervised metric learning based on WMD, called Supervised WMD (S-WMD). S-WMD transforms word embedding vectors and re-weights the bag-of-words via supervised learning.

To solve the optimal transport problem, linear programming can be used. However, using linear programming requires cubic time with respect to the number of coordinates (Pele & Werman, 2009). To reduce this time complexity, Cuturi (2013) proposed the entropic regularized optimal transport, which is called the Sinkhorn algorithm and can be solved in quadratic time.

Tree-Wasserstein Distances: Given a tree $\mathcal{T} = (\mathbf{V}, \mathbf{E})$ rooted at v_1 with non-negative edge lengths, the tree metric $d_{\mathcal{T}}$ between two nodes is the total length of the path between the nodes. Let $\Gamma(v)$ be a set of nodes contained in the subtree of $\mathcal{T}$ rooted at $v \in \mathbf{V}$. For all $v \in \mathbf{V} \setminus \{v_1\}$, there exists a unique node $u \in \mathbf{V}$ which is the parent node of v and we write w_v for the length of the edge from v to its parent node. Given two measures μ and ν supported on $\mathcal{T}$, the tree-Wasserstein distance between μ and ν is calculated as follows:

$$W_{d_{\mathcal{T}}}(\mu, \nu) = \sum_{v \in \mathbf{V} \setminus \{v_1\}} w_v \left| \mu(\Gamma(v)) - \nu(\Gamma(v)) \right|. \quad (1)$$

The parent node of the root v_1 does not exist, and the length of the edge w_{v_1} is not defined. However, because $\mu(\Gamma(v_1)) = \nu(\Gamma(v_1)) = 1$, we define $w_{v_1} = 1$ for simplicity; the tree-Wasserstein distance can be written as $W_{d_{\mathcal{T}}}(\mu, \nu) = \sum_{v \in \mathbf{V}} w_v |\mu(\Gamma(v)) - \nu(\Gamma(v))|$. The key property of the tree-Wasserstein distance is that it can be computed in linear time with respect to the number of nodes. Furthermore, the tree-Wasserstein distance between μ and ν is regarded as the L1 distance between their corresponding $|\mathbf{V}|$-dimensional vectors whose elements corresponding to v are $w_v \mu(\Gamma(v))$ and $w_v \nu(\Gamma(v))$. In practice, these embedding vectors are sparse. This allows for faster implementation (Backurs et al., 2020). In the unbalanced setting, Sato et al. (2020) proposed a method to compute the tree-Wasserstein distance in quasi-linear time.

To compute the tree-Wasserstein distance, we need to construct a tree metric. Indyk & Thaper (2003) proposed a method to embed the coordinates into the tree metric in the context of image retrieval, which is called Quadtree. Le et al. (2019) proposed the tree-sliced Wasserstein (TSW) distance, which is a variant of the sliced-Wasserstein distance (Rabin et al., 2011; Kolouri et al., 2018; 2019a; Deshpande et al., 2019). The TSW distance is the average of the tree-Wasserstein distances on the sampled tree metrics. Recently, Backurs et al. (2020) proposed Flowtree, which computes the optimal flow on Quadtree, then computes the cost of the optimal flow on the ground metric, unlike Quadtree and the TSW distance. Flowtree is slower than Quadtree in computing the optimal flow, but can theoretically approximate the Wasserstein distance more accurately. These previous works aimed to approximate the Wasserstein distance on the Euclidean metric with the tree-Wasserstein distance. In contrast to these previous works, our goal is not to approximate the ground metric, but to construct a tree metric that represents the task-specific distance by leveraging the label information of the documents; so that the tree-Wasserstein distance between documents with the same label is small, and the tree-Wasserstein distance between documents with different labels is large.

2.2. Continuous Optimization for a Tree

When solving the task of learning a tree structure as a continuous optimization problem, learning in hyperbolic space is highly related. Hyperbolic space has a property that is similar to that of a tree, where the volume increases exponentially with the radius, and the number of nodes increases exponentially with the depth of the tree. Using this property, various methods that solve continuous optimization to learn a tree structure by representing the nodes with coordinates in hyperbolic space have been proposed (Nickel & Kiela, 2017; Ganea et al., 2018). In hierarchical clustering, Monath et al. (2019); Chami et al. (2020) formulated the probability or the coordinates of the lowest common ancestors in hyperbolic space and constructed a tree by minimizing a soft variant of Dasgupta's cost (Dasgupta, 2016), which is the well-known cost for hierarchical clustering. However, these methods are not applicable to the tree-Wasserstein distance because it is necessary to formulate whether a node is contained in a subtree (i.e., $\Gamma(v)$). In contrast to these works, we introduce the conditions of an adjacency matrix to be the adjacency matrix of a tree, formulate the probability that a node is contained in a subtree, and then propose a continuous optimization problem with respect to the adjacency matrix.

3. Proposed Method

In this section, we first introduce a soft variant of the tree-Wasserstein distance; then we propose the STW distance.

3.1. Problem Setting

We have a finite size vocabulary set $\boldsymbol{Z} = \{z_1, z_2, \ldots, z_{N_{\text{leaf}}}\}$ consisting of N_{leaf} words and a training dataset $\mathcal{D} = \{(\mathbf{a}_i, y_i)\}_{i=1}^M$ where N_{leaf}-dimensional vector $\mathbf{a}_i = (a_i^{(1)}, a_i^{(2)}, \ldots, a_i^{(N_{\text{leaf}})})^\top \in [0, 1]^{N_{\text{leaf}}}$ is the normalized bag-of-words (i.e., $\mathbf{a}_i^\top \mathbf{1}_{N_{\text{leaf}}} = 1$), and $y_i \in \mathbb{N}$ is a label of document i. In the following sections, we assign words to leaf nodes of the tree, as in Quadtree and the TSW distance. We refer to the nodes corresponding to each word as *leaf* nodes and the nodes not corresponding to any word as *internal* nodes. Note that leaf nodes have no child nodes, but there may be internal nodes that do not have child nodes. To construct the tree metric by leveraging the label information of documents, assume that we have a set of nodes $\boldsymbol{V} = \{v_1, v_2, \ldots, v_N\}$, in which v_1 is the root. We consider constructing the tree metric by learning the parent–child relationships of these nodes. Let N_{in} be the number of internal nodes ($N = N_{\text{in}} + N_{\text{leaf}}$). $\boldsymbol{V}_{\text{in}} = \{v_1, v_2, \ldots, v_{N_{\text{in}}}\}$ is a set of internal nodes. $\boldsymbol{V}_{\text{leaf}} = \{v_{N_{\text{in}}+1}, \ldots, v_N\}$ is a set of leaf nodes. w_v is the length of an edge from v to the parent node of v. For simplicity, we define $w_{v_1} = 1$. We assume that the word z_i corresponds to $v_{N_{\text{in}}+i}$. We denote the training dataset using the discrete measure $\mathcal{D} = \{(\mu_i, y_i)\}_{i=1}^M$, where $\mu_i = \sum_j a_i^{(j)} \delta(v_{N_{\text{in}}+j}, \cdot)$ is the discrete measure that represents the document i.

3.2. Soft Tree-Wasserstein Distance

Our goal is to construct a tree metric such that the tree-Wasserstein distance between documents with the same label is small and the distance between documents with different labels is large. To achieve this, we first show the conditions of the parent–child relationships of a tree, formulate the probability that a node is contained in a subtree using these conditions, and then propose a soft variant of the tree-Wasserstein distance.

The parent–child relationships of a tree with a specific root can be represented by the adjacency matrix of the directed tree, which has edges from child nodes to their parent nodes. We show the conditions for an adjacency matrix to be an adjacency matrix of a tree.

Theorem 1. *If the adjacency matrix $\mathbf{D}_{par} \in \{0, 1\}^{N \times N}$ of a directed graph $G = (\boldsymbol{V}, \boldsymbol{E})$ satisfies the following conditions:*

(1) $\mathbf{D}_{par}$ is a strictly upper triangular matrix.

(2) $\mathbf{D}_{par}^\top \mathbf{1}_N = (0, 1, \cdots, 1)^\top$.

then G is a directed tree with v_1 as the root.

Appendix details the proof. To introduce a soft variant of the tree-Wasserstein distance, we relax $\mathbf{D}_{\text{par}} \in \{0, 1\}^{N \times N}$

to $\mathbf{D}_{\text{par}} \in [0,1]^{N \times N}$ while satisfying the conditions of Theorem 1. In $\mathbf{D}_{\text{par}}$, the elements in the first column are all zero; in the second and subsequent columns, the sum of the elements in each column is one. In other words, the element in the i-th row and j-th column of $\mathbf{D}_{\text{par}}$ is the probability that v_i is a parent of v_j. The elements in the i-th row and j-th column of $\mathbf{D}_{\text{par}}^k$ denotes the probability that there exists a path from v_j to v_i with k steps. The element in the i-th row and j-th column of the sum of the infinite geometric series is the probability that there exists a path from v_j to v_i. In other words, it means the probability that v_j is contained in the subtree rooted at v_i. We refer to this probability as $P_{\text{sub}}(v_j|v_i)$ and define it as follows:

$$P_{\text{sub}}(v_j|v_i) = \left[\sum_{k=0}^{\infty} \mathbf{D}_{\text{par}}^k \right]_{i,j} = \left[(\mathbf{I} - \mathbf{D}_{\text{par}})^{-1} \right]_{i,j}. \quad (2)$$

$\mathbf{D}_{\text{par}}$ is a nilpotent matrix because it is an upper triangular matrix and all the diagonal elements are zero. Therefore, the sum of the infinite geometric series converges to $(\mathbf{I} - \mathbf{D}_{\text{par}})^{-1}$. We show more details in the Appendix. By using this probability, we define the *soft tree-Wasserstein distance* $W_{d_{\mathcal{T}}}^{\text{soft}}(\mu_i, \mu_j)$ as follows:

$$W_{d_{\mathcal{T}}}^{\text{soft}}(\mu_i, \mu_j)$$
$$= \sum_{v \in \mathbf{V}} w_v \left| \sum_{x \in V_{\text{leaf}}} P_{\text{sub}}(x|v) \left(\mu_i(x) - \mu_j(x) \right) \right|_\alpha, \quad (3)$$

where $|\cdot|_\alpha$ is a smooth approximation of the L1 norm, defined as follows:

$$|x|_\alpha = \frac{x(e^{\alpha x} - e^{-\alpha x})}{2 + e^{\alpha x} + e^{-\alpha x}}.$$

It has been shown that if α approaches ∞, then $|\cdot|_\alpha$ converges to the L1 norm (Lange et al., 2014). Other differentiable approximations for the L1 norm can also be used. The soft tree-Wasserstein distance satisfies the identity of indiscernibles and the symmetry, but does not satisfy the triangle inequality, because $|\cdot|_\alpha$ does not satisfy the triangle inequality. Thus, the soft tree-Wasserstein distance is not a metric. However, the soft tree-Wasserstein distance satisfies the following theorem; the proof is shown in the Appendix.

Theorem 2. *If the tree metric is given and α approaches ∞, then the soft tree-Wasserstein distance converges to the tree-Wasserstein distance.*

3.3. Fast Computation Method

Because the size of $\mathbf{D}_{\text{par}}$ is large, calculating the inverse matrix in Eq. (2) has high computational cost and memory consumption. Next, we introduce a method to reduce this cost by utilizing the property of $\mathbf{D}_{\text{par}}$.

We arranged the index of nodes such that the index of an internal node was less than the index of a leaf node. As pointed out earlier, leaf nodes have no child nodes. Then, the lower block of $\mathbf{D}_{\text{par}}$ is a zero matrix and $\mathbf{D}_{\text{par}}$ can be partitioned into four blocks as follows:

$$\mathbf{D}_{\text{par}} = \begin{pmatrix} \mathbf{D}_1 & \mathbf{D}_2 \\ \mathbf{0} & \mathbf{0} \end{pmatrix}, \quad (4)$$

where $\mathbf{D}_1$ is an $N_{\text{in}} \times N_{\text{in}}$ matrix, and $\mathbf{D}_2$ is an $N_{\text{in}} \times N_{\text{leaf}}$ matrix. $\mathbf{D}_1$ denotes the parent–child relationships of a tree consisting of internal nodes, and $\mathbf{D}_2$ represents which internal nodes the leaf nodes connect to. Utilizing this property and the constraints of $\mathbf{D}_{\text{par}}$, we can calculate the inverse matrix as follows:

$$(\mathbf{I} - \mathbf{D}_{\text{par}})^{-1} = \begin{pmatrix} (\mathbf{I} - \mathbf{D}_1)^{-1} & (\mathbf{I} - \mathbf{D}_1)^{-1}\mathbf{D}_2 \\ \mathbf{0} & \mathbf{I} \end{pmatrix}, \quad (5)$$

where $\mathbf{I} - \mathbf{D}_1$ is a regular matrix, and there exists an inverse matrix because $\mathbf{D}_1$ is an upper triangular matrix, and all diagonal elements are zero. The bottom two blocks do not need to be retained because they are not learned and we can reduce the memory consumption. Since N_{in} is, in general, 150 to 4000, the computation of the inverse matrix $(\mathbf{I} - \mathbf{D}_1)^{-1}$ is not expensive. Thus, we can reduce the computational cost and memory consumption.

3.4. Supervised Tree-Wasserstein Distancce

Our goal is to construct a tree metric such that the tree-Wasserstein distance between documents with the same label is small and the tree-Wasserstein distance between documents with different labels is large. To achieve this, we use a contrastive loss similar to prior works (Hadsell et al., 2006) as follows:

$$\mathcal{L}(\mathbf{D}_{\text{par}}, \mathbf{w}_v) = \frac{1}{|\mathcal{D}_p|} \sum_{(i,j) \in \mathcal{D}_p} W_{d_{\mathcal{T}}}^{\text{soft}}(\mu_i, \mu_j)$$
$$- \frac{1}{|\mathcal{D}_n|} \sum_{(i,j) \in \mathcal{D}_n} \min \left\{ W_{d_{\mathcal{T}}}^{\text{soft}}(\mu_i, \mu_j), m \right\},$$

where $\mathbf{w}_v = (w_{v_1}, w_{v_2}, \cdots, w_{v_N})^\top$ is an N-dimensional vector, $\mathcal{D}_p = \{(i,j)|y_i = y_j\}$ is a set of index pairs of documents that have the same label, $\mathcal{D}_n = \{(i,j)|y_i \neq y_j\}$ is a set of index pairs of documents that have different labels, and m is the margin.

However, it is difficult to minimize this loss function with respect to $\mathbf{D}_{\text{par}}$ and $\mathbf{w}_v$ because the joint optimization of $\mathbf{D}_1$, $\mathbf{D}_2$, and $\mathbf{w}_v$ has too many degrees of freedom. To solve this problem, we propose initializing $\mathbf{D}_1$ as an adjacency matrix of a tree consisting of internal nodes and $\mathbf{w}_v = \mathbf{1}_N$, fix $\mathbf{D}_1$ and $\mathbf{w}_v$ at the initial value, and minimize the loss with respect to only $\mathbf{D}_2$. In other words, given a tree $\mathcal{T}' =$

$(V_{\text{in}}, E_{\text{in}})$ whose adjacency matrix is $\mathbf{D}_1$ and edge lengths are all one, we optimize where to connect leaf nodes to $\mathcal{T}'$. As a by-product, the inverse matrix in Eq. (5) needs to be calculated only once before training. To optimize the loss function while satisfying the conditions of Theorem 1, we propose to calculate $\mathbf{D}_2$ using the softmax function as follows:

$$[\mathbf{D}_2]_{i,j} = \frac{\exp\left([\mathbf{\Theta}]_{i,j}\right)}{\sum_{i'=1}^{N_{\text{in}}} \exp\left([\mathbf{\Theta}]_{i',j}\right)},$$

where $\mathbf{\Theta} \in \mathbb{R}^{N_{\text{in}} \times N_{\text{leaf}}}$ is the parameter to be optimized. Using the softmax function, $\mathbf{D}_2^\top \mathbf{1}_{N_{\text{in}}} = \mathbf{1}_{N_{\text{leaf}}}$ and $\mathbf{D}_1$ is initialized such that $\mathbf{D}_1^\top \mathbf{1}_{N_{\text{in}}} = (0, 1, \cdots, 1)^\top$; then $\mathbf{D}_1$ and $\mathbf{D}_2$ satisfy the conditions of Theorem 1. Note that other softmax-like functions can also be used (Martins & Astudillo, 2016; Kong et al., 2020) as long as the constraint that the sum is one is satisfied. In summary, our optimization problem is given as follows:

$$\min_{\mathbf{\Theta} \in \mathbb{R}^{N_{\text{in}} \times N_{\text{leaf}}}} \mathcal{L}(\mathbf{D}_{\text{par}}, \mathbf{w}_v), \tag{6}$$

where $\mathbf{D}_1$ is fixed at initial values and $\mathbf{w}_v = \mathbf{1}_N$. Since this objective function is differentiable with respect to $\mathbf{\Theta}$, we can optimize it by stochastic gradient descent. After optimization, for each leaf node, we select one of the most probable parents and construct the tree metric:

$$\mathbf{D}_2^* = (\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_{N_{\text{leaf}}}) \in \{0, 1\}^{N_{\text{in}} \times N_{\text{leaf}}},$$

where $\mathbf{e}_j \in \{0, 1\}^{N_{\text{in}}}$ is the one-hot vector whose $k^* = \text{argmax}_k [\mathbf{D}_2]_{k,j}$ th element is one and the other elements are zero. We substitute $\mathbf{D}_2^*$ and $\mathbf{D}_1$ in Eq. (4) and obtain the tree metric that represents the task-specific distance. We refer to this approach as the *Supervised Tree-Wasserstein* (STW) distance.

The tree-Wasserstein distance between μ_i and μ_j can be considered as the L1 distance between their corresponding vectors. Using the formulation of the soft tree-Wasserstein distance, the tree-Wasserstein distance can be computed as the L1 norm of the following vector:

$$\mathbf{w}_v \circ \left\{ \begin{pmatrix} (\mathbf{I} - \mathbf{D}_1)^{-1} & (\mathbf{I} - \mathbf{D}_1)^{-1}\mathbf{D}_2^* \\ \mathbf{0} & \mathbf{I} \end{pmatrix} \begin{pmatrix} \mathbf{0}_{N_{\text{in}}} \\ \mathbf{a}_i - \mathbf{a}_j \end{pmatrix} \right\},$$

where $\circ$ is the element-wise Hadamard product. As can be seen above, this formulation can be generalized to the case of comparing one document $\mathbf{a}_1$ with $M - 1$ documents $\mathbf{a}_2, \mathbf{a}_3, \ldots, \mathbf{a}_M$. Then $M - 1$ documents can be compared simultaneously by replacing the right vector in the above equation with $\begin{pmatrix} \mathbf{0}_{N_{\text{in}}} & \cdots & \mathbf{0}_{N_{\text{in}}} \\ \mathbf{a}_2 - \mathbf{a}_1 & \cdots & \mathbf{a}_M - \mathbf{a}_1 \end{pmatrix}$. Therefore, the STW distance can be computed on a GPU and can compare multiple documents simultaneously.

Algorithm 1 Implementation of the STW distance, using PyTorch syntax.

1: **Input:** normalized bag-of-words $\mathbf{a}_i, \mathbf{a}_j, \mathbf{w}_v = \mathbf{1}_N$.
2: **Output:** tree-Wasserstein distance between $\mathbf{a}_i$ and $\mathbf{a}_j$.
3: $\mathbf{a} = \mathbf{a}_i - \mathbf{a}_j$
4: $\mathbf{A} = (\mathbf{I} - \mathbf{D}_1)^{-1}$
5: $\mathbf{D}_2 = \text{softmax}(\mathbf{\Theta}, \text{dim=0})$
6: $\mathbf{D}_2^* = \mathbf{D}_2.\text{ge}(\mathbf{D}_2.\text{max}(0, \text{keepdim=True})[0]).\text{float}()$
7: $\mathbf{C} = \text{mm}(\mathbf{A}, \mathbf{D}_2^*)$
8: **return** $\text{abs}(\text{mv}(\mathbf{C}, \mathbf{a})).\text{sum}() + \text{abs}(\mathbf{a}).\text{sum}()$

3.5. Implementation Details

We initialize $\mathbf{D}_1$ such that the tree $\mathcal{T}'$ with this adjacency matrix is a perfect k-ary tree of depth d. We show the pseudo-code of the STW distance for inference in Algorithm 1. In practice, lines 4–7 need to be computed only once before inference. During training, we skip line 6, use the approximation of the L1 norm in line 8, compute the loss, and update the parameter $\mathbf{\Theta}$. Since all operations can run on a GPU and are differentiable, we can optimize $\mathbf{\Theta}$ using backpropagation and mini-batch stochastic gradient descent. This can be easily extended to an implementation that is suitable for batch processing. We found that when the number of unique words contained in a document is large, the optimization is difficult because the elements of the normalized bag-of-words reach zero. To address this issue, we multiply a fixed value 5 to $\mathbf{a}$ in Algorithm 1 during training.

For all $v_i \in V_{\text{in}}$ and $v_j \in V_{\text{leaf}}$, the number of nodes contained in a path from v_j to v_i is at most $d + 2$. If a node $v_{j+N_{\text{in}}}$ is contained in the subtree rooted at v_i, then $[\mathbf{C}]_{i,j}$ is one, and is zero otherwise. Therefore, $\mathbf{C}$ is a sparse matrix that has at most $(d + 1) \times N_{\text{leaf}}$ non-zero elements, and $\mathbf{a}$ is a sparse vector because $s \ll N_{\text{leaf}}$, where s denotes the number of unique words contained in the two documents to be compared. In general, since GPUs are not suitable for multiplications of sparse matrices, it is faster to compute them as multiplications of dense matrices when computing on a GPU. In the following experiments, we evaluate the STW distance on a GPU as multiplications of dense matrices. However, when run on a CPU, it can be computed in $O(sd)$ by using this sparsity.

4. Experimental Results

We evaluate the following methods in document classification tasks on the synthetic and six real datasets following S-WMD in the test error rate of the k-nearest neighbors (kNN) and the time consumption: TWITTER, AMAZON, CLASSIC, BBCSPORT, OHSUMED, and REUTERS. Datasets are split into train/test as with the previous works (Kusner et al., 2015; Huang et al., 2016). Table 1 lists the

Table 1. Datasets used for the experiments.

	BOW DIMENSION	AVERAGE WORDS
TWITTER	6344	9.9
CLASSIC	24277	38.6
AMAZON	42063	45.0
BBCSPORT	13243	117
OHSUMED	31789	59.2
REUTERS	22425	37.1

number of unique words contained in the dataset (bag-of-words dimension) and the average number of unique words contained in a document for all real datasets.

4.1. Baseline Methods

Word Mover's Distance (WMD) (Kusner et al., 2015): The document metric formulated by the optimal transport problem, as described in Section 2.

Supervised Word Mover's Distance (S-WMD) (Huang et al., 2016): Supervised metric learning based on WMD.

Quadtree (Indyk & Thaper, 2003): To construct the tree metric, we first obtain a randomly shifted hypercube containing all word embedding vectors. Next, we recursively divide the hypercube into hypercubes with half side length until there is only one word embedding vector in the hypercube. Each hypercube corresponds to a node, which has child nodes that correspond to hypercubes with half side length created by the split. The tree constructed in this way is called Quadtree. After constructing Quadtree, we compute the tree-Wasserstein distance in Eq. (1).

Flowtree (Backurs et al., 2020): Flowtree computes the transport plan on Quadtree, and then computes the cost on the ground metric.

Tree-Sliced Wasserstein (TSW) Distance (Le et al., 2019): The TSW distance samples the tree metrics, and then computes the average distance of tree-Wasserstein distances on these tree metrics. A previous work (Le et al., 2019) showed that increasing the sampling size results in higher accuracy, but requires more computation time, and recommended 10 samples. Following this, we evaluated the TSW distance with the deepest level of the tree of 6 and the number of child nodes of 5 with sampling numbers of 1, 5, and 10. For sampling size, we refer to TSW-1, TSW-5, and TSW-10, respectively.

Supervised Tree-Wasserstein (STW) Distance: We initialize $\mathbf{D}_1$ such that the tree whose adjacency matrix is $\mathbf{D}_1$ is a perfect 5-ary tree of depth 5, and optimize Eq. (6) using Adam (Kingma & Ba, 2015) and LARS (You et al., 2017). After optimization, the deepest level of the tree is 5 or 6. To select the margin m, we use 20% of the training dataset for validation. We then train our model at a learning rate of 0.1 and a batch size of 100 for 30 epochs. To avoid overfitting,
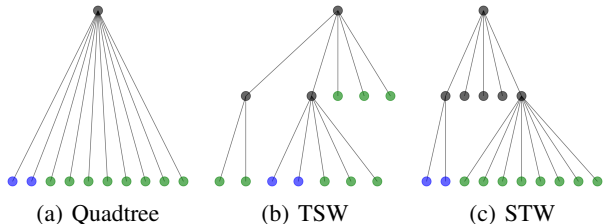

Figure 1. Trees constructed by Quadtree, the TSW distance, and the STW distance on the synthetic dataset. Flowtree computes the optimal flow on Quadtree. Nodes that correspond to internal nodes are black-filled; nodes that correspond to the words "piano" and "violin" are blue-filled; and others are green-filled.

Table 2. The kNN test error rate on the synthetic dataset.

QUADTREE	FLOWTREE	TSW-1/5/10	STW
0.3	1.6	7.5 / 4.2 / 3.9	0.0

we evaluated the STW distance using the parameters with the lowest loss in 30 epochs of the validation dataset.

4.2. Experimental Setup

We use word2vec (Mikolov et al., 2013), which is pre-trained on Google News [1] as the word embedding vectors for WMD, S-WMD, Quadtree, Flowtree, and the TSW distance. For measuring the time consumption, we use the public implementation [2] of (Kusner et al., 2015) for WMD and the public implementation [3] of (Backurs et al., 2020), which is written in C++ and Python, for Quadtree and Flowtree. We implement S-WMD, and the TSW and STW distances in PyTorch. The public implementation of WMD is written in C and Python and uses the algorithm developed by (Pele & Werman, 2009), which requires cubic time. Additionally, we implement WMD with Sinkhorn algorithm in PyTorch, which we refer to as WMD (Sinkhorn). The parameter of the Sinkhorn algorithm for WMD (Sinkhorn) and our implementation of S-WMD is same as the public implementation [4] of (Huang et al., 2016). We evaluated WMD (Sinkhorn), S-WMD, and the TSW and STW distances on Nvidia Quadro RTX 8000, and WMD, Quadtree, and Flowtree on Intel Xeon CPU E5-2690 v4 (2.60 GHz).

4.3. Results on the Synthetic Dataset

By using the synthetic dataset, we first show that the STW distance can construct a tree metric that represents a task-specific distance and improves the accuracy of the document classification task. We generated the synthetic dataset so that documents consist of only ten words: "piano," "violin,"

---

[1] https://code.google.com/p/word2vec
[2] https://github.com/mkusner/wmd
[3] https://github.com/ilyaraz/ot_estimators
[4] https://github.com/gaohuang/S-WMD

Table 3. The kNN test error for real datasets. WMD and S-WMD give the results from (Huang et al., 2016).

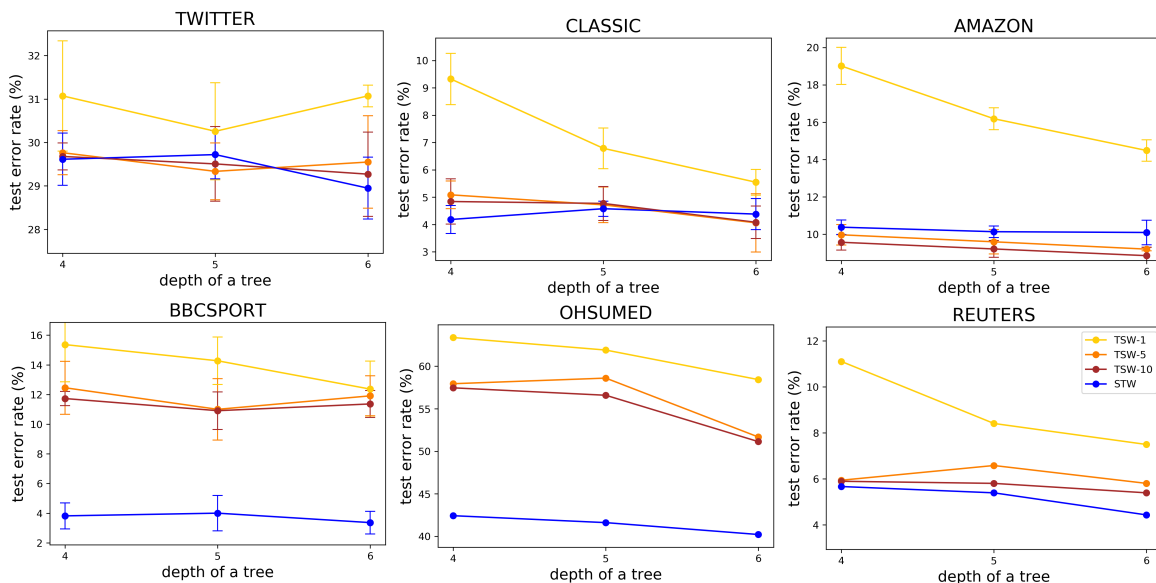|	TWITTER	AMAZON	CLASSIC	BBCSPORT	OHSUMED	REUTERS
WMD	28.7 ± 0.6	7.4 ± 0.3	2.8 ± 0.1	4.6 ± 0.7	44.5	3.5
S-WMD	27.5 ± 0.5	5.8 ± 0.1	3.2 ± 0.2	2.1 ± 0.5	34.3	3.2
QUADTREE	30.4 ± 0.8	10.7 ± 0.3	4.1 ± 0.4	4.5 ± 0.5	44.0	5.2
FLOWTREE	29.8 ± 0.9	9.9 ± 0.3	5.6 ± 0.6	4.7 ± 1.1	44.4	4.7
TSW-1	30.2 ± 1.3	14.5 ± 0.6	5.5 ± 0.5	12.4 ± 1.9	58.4	7.5
TSW-5	29.5 ± 1.1	9.2 ± 0.1	4.1 ± 0.4	11.9 ± 1.3	51.7	5.8
TSW-10	29.3 ± 1.0	8.9 ± 0.5	4.1 ± 0.6	11.4 ± 0.9	51.1	5.4
STW	28.9 ± 0.7	10.1 ± 0.7	4.4 ± 0.7	3.4 ± 0.8	40.2	4.4

Figure 2. The kNN test error rate on real datasets when varying the depth level of the tree. For the STW distance, if the tree consisting of internal nodes is initialized so that its depth is d, the depth of the tree after optimization is d or $d + 1$. In this figure, when the depth of the tree consisting of internal nodes is initialized such that its depth is d, the depth of the STW distance is considered to be $d + 1$.

"cello," "viola," "contrabass," "trumpet," "trombone," "clarinet," "flute," and "harpsichord." Each word contains zero or one and documents are classified into two classes based on whether the word "piano" or "violin" is contained. We initialize $\mathbf{D}_1$ so that the tree whose adjacency matrix is $\mathbf{D}_1$ is a perfect 5-ary tree of depth 1 for easy visualization.

We show the trees constructed by Quadtree, Flowtree, the TSW and STW distances in Figure 1 and the kNN test error rate in Table 2. Quadtree constructs a tree so that the distance between all words is the same because the dimension of the word embedding vector is high and each word is assigned to a different hypercube. The TSW distance constructs a tree so that the words "piano" and "violin" are not far from other words. However, the STW distance constructs a tree so that the words "piano" and "violin" are close and far from other words, and the words except for the words "piano" and "violin" are close together. As a result, the STW distance outperforms Quadtree, Flowtree, and the TSW distance.

4.4. Results on Real Datasets

We first discuss the accuracy of document classification tasks on real datasets, and then discuss the time consumption to compute the distances. We list the kNN test error rates in Table 3. On TWITTER, BBCSPORT, OHSUMED, and REUTERS, the STW distance outperforms Quadtree, Flowtree, and the TSW distance. On AMAZON and CLASSIC, the STW distance outperforms the TSW-1 distance and is competitive with Quadtree, Flowtree, the TSW-5 distance, and the TSW-10 distance, respectively. In particular, the error rate of the TSW distance is approximately 10% higher than that of WMD on BBCSPORT and OHSUMED, but the STW distance improves the error rate and outperforms WMD. On the other hand, the STW distance still underperforms WMD in other datasets and all tree-based methods underperform S-WMD in all datasets.

To construct the tree metric in the TSW and STW distances, we need to set the depth level of the tree as the hyperparameters. We evaluated how the tree's depth level affects
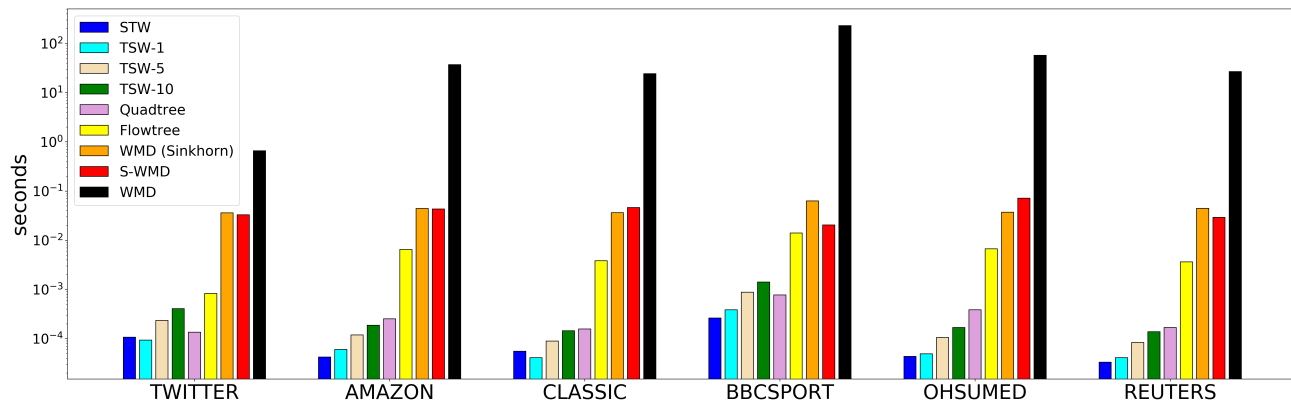
Figure 3. Average time consumption for comparing 500 documents with one document. For the STW distance and the TSW distance, the batch size is set to the number of documents contained in the training dataset. For WMD (Sinkhorn) and S-WMD, the batch size is set to 500 due to the memory size limitations. To obtain the average time consumption, we sample 100 documents as queries and measure the time consumption.
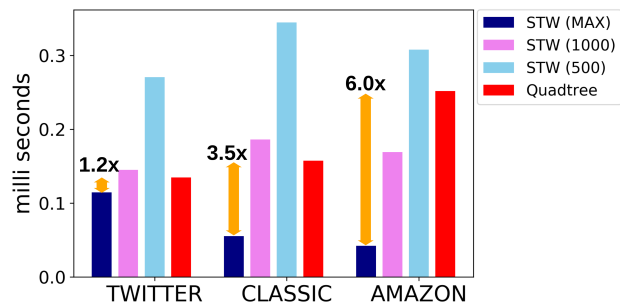

Figure 4. Average time consumption to compare one document with 500 documents. The number in the bracket indicates the batch size and *MAX* means the number of documents contained in the training dataset.

the accuracy of the TSW and STW distances. In Figure 2, we show the kNN test error rate when the STW distance is initialized, such that $\mathbf{D}_1$ is an adjacency matrix of the depth level of trees 3, 4, and 5, and the TSW distance is sampled so that the depth level of the tree is 4, 5, and 6. The results show that, in general, the deeper the depth level of the tree, the higher the accuracy. When the depth level of the tree is 4, the accuracy of the TSW-1 distance is considerably worse than when the depth level of the tree is 6, whereas the STW distance is only approximately 2% worse. The results indicate that the STW distance is more accurate than the TSW-1 distance, especially when the tree is shallow.

Next, we discuss the average time consumption to calculate distance. We show the time required to compare 500 documents with one document in Figure 3. Quadtree, Flowtree, and the TSW and STW distances are faster than WMD, WMD (Sinkhorn), and S-WMD on all datasets. The TSW-10 distance calculates the tree-Wasserstein distance 10 times, which is approximately 10 times slower than Quadtree, and the TSW-1 and STW distances. The public implementation of Quadtree uses an algorithm that is suitable for CPUs, which runs in linear time with respect to the number of

unique words in the document. The time complexity of the implementation of the STW distance depends on the number of unique words in the dataset, but runs on a GPU and is suitable for batch processing. Therefore, when comparing a large number of documents, our algorithm is more efficient than the existing algorithm for computing the tree-Wasserstein distance. In Figure 4, we show the average time consumption when varying the batch sizes on TWITTER, CLASSIC, and AMAZON for Quadtree and the STW distance. The results indicate that, if the batch size is sufficiently large, the STW distance is faster than Quadtree. In particular, on AMAZON, when the batch size is set to the number of documents contained in the training dataset, the STW distance is about six times faster than Quadtree. Additional experiments when varying the batch size are included in the Appendix.

5. Conclusion

In this work, we proposed the soft tree-Wasserstein distance and the supervised tree-Wasserstein distance. The soft tree-Wasserstein distance is differentiable with respect to the probability of the parent–child relationships of a tree and is formulated only by matrix multiplications. By using the soft tree-Wasserstein distance, we formulated the STW distance as a continuous optimization problem, which is end-to-end trainable and constructs the tree metric by leveraging the label information of documents. Through the experiments on the synthetic and real datasets, we showed that the STW distance can be computed quickly and can improve the accuracy of document classification tasks. Furthermore, because the STW distance is suitable for batch processing, it is more efficient than existing methods for computing the Wasserstein distance, especially when comparing a large number of documents.

Acknowledgement

References

Atasu, K. and Mittelholzer, T. Linear-complexity data-parallel earth mover's distance approximations. In *International Conference on Machine Learning*, 2019.

Backurs, A., Dong, Y., Indyk, P., Razenshteyn, I., and Wagner, T. Scalable nearest neighbor search for optimal transport. In *International Conference on Machine Learning*, 2020.

Chami, I., Gu, A., Chatziafratis, V., and Re, C. From trees to continuous embeddings and back: Hyperbolic hierarchical clustering. In *Advances in Neural Information Processing Systems*, 2020.

Cuturi, M. Sinkhorn distances: Lightspeed computation of optimal transport. In *Advances in Neural Information Processing Systems*, 2013.

Dasgupta, S. A cost function for similarity-based hierarchical clustering. In *ACM Symposium on Theory of Computing*, 2016.

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 *IEEE conference on Computer Vision and Pattern Recognition*, 2019.

Ganea, O., Becigneul, G., and Hofmann, T. Hyperbolic entailment cones for learning hierarchical embeddings. In *International Conference on Machine Learning*, 2018.

Hadsell, R., Chopra, S., and LeCun, Y. Dimensionality reduction by learning an invariant mapping. In *IEEE conference on Computer Vision and Pattern Recognition*, 2006.

Huang, G., Guo, C., Kusner, M. J., Sun, Y., Sha, F., and Weinberger, K. Q. Supervised word mover's distance. In *Advances in Neural Information Processing Systems*, 2016.

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

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

Kolouri, S., Rohde, G. K., and Hoffmann, H. Sliced wasserstein distance for learning gaussian mixture models. In *IEEE conference on Computer Vision and Pattern Recognition*, 2018.

Kolouri, S., Nadjahi, K., Simsekli, U., Badeau, R., and Rohde, G. Generalized sliced wasserstein distances. In *Advances in Neural Information Processing Systems*, 2019a.

Kolouri, S., Pope, P. E., Martin, C. E., and Rohde, G. K. Sliced wasserstein auto-encoders. In *International Conference on Learning Representations*, 2019b.

Kong, W., Krichene, W., Mayoraz, N., Rendle, S., and Zhang, L. Rankmax: An adaptive projection alternative to the softmax function. In *Advances in Neural Information Processing Systems*, 2020.

Korte, B. and Vygen, J. *Combinatorial Optimization: Theory and Algorithms*. Springer, 3rd edition, 2006.

Kusner, M. J., Sun, Y., Kolkin, N. I., and Weinberger, K. Q. From word embeddings to document distances. In *International Conference on Machine Learning*, 2015.

Lange, M., Zühlke, D., Holz, O., and Villmann, T. Applications of lp-norms and their smooth approximations for gradient based learning vector quantization. In *European Symposium on Artificial Neural Networks*, 2014.

Le, T., Yamada, M., Fukumizu, K., and Cuturi, M. Tree-sliced variants of wasserstein distances. In *Advances in Neural Information Processing Systems*, 2019.

Liu, Y., Zhu, L., Yamada, M., and Yang, Y. Semantic correspondence as an optimal transport problem. In *IEEE conference on Computer Vision and Pattern Recognition*, 2020.

Martins, A. and Astudillo, R. From softmax to sparsemax: A sparse model of attention and multi-label classification. In *International Conference on Machine Learning*, 2016.

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*, 2013.

Monath, N., Zaheer, M., Silva, D., McCallum, A., and Ahmed, A. Gradient-based hierarchical clustering using continuous representations of trees in hyperbolic space. In *International Conference on Knowledge Discovery and Data Mining*, 2019.

Nickel, M. and Kiela, D. Poincaré embeddings for learning hierarchical representations. In *Advances in Neural Information Processing Systems*, 2017.

Pele, O. and Werman, M. Fast and robust earth mover's distances. In *IEEE conference on International Conference on Computer Vision*, 2009.

Rabin, J., Peyré, G., Delon, J., and Bernot, M. Wasserstein barycenter and its application to texture mixing. In *Scale Space and Variational Methods in Computer Vision*, 2011.

Sarlin, P., DeTone, D., Malisiewicz, T., and Rabinovich, A. Superglue: Learning feature matching with graph neural networks. In *IEEE conference on Computer Vision and Pattern Recognition*, 2020.

Sato, R., Yamada, M., and Kashima, H. Fast unbalanced optimal transport on a tree. In *Advances in Neural Information Processing Systems*, 2020.

You, Y., Gitman, I., and Ginsburg, B. Large batch training of convolutional networks. *arXiv preprint arXiv:1708.03888*, 2017.

Yurochkin, M., Claici, S., Chien, E., Mirzazadeh, F., and Solomon, J. M. Hierarchical optimal transport for document representation. In *Advances in Neural Information Processing Systems*, 2019.