FASTER ALGORITHMS FOR MACHINE LEARNING PROBLEMS IN HIGH DIMENSION

By

Mingquan Ye

A THESIS

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

Computer Science – Master of Science

2019

ABSTRACT

FASTER ALGORITHMS FOR MACHINE LEARNING PROBLEMS IN HIGH DIMENSION

By

Mingquan Ye

When dealing with datasets with high dimension, the existing machine learning algorithms often do not work in practice. Actually, most of the real-world data has the nature of low intrinsic dimension. For example, data often lies on a low-dimensional manifold or has a low doubling dimension. Inspired by this phenomenon, this thesis tries to improve the time complexities of two fundamental problems in machine learning using some techniques in computational geometry.

In Chapter two, we propose a bi-criteria approximation algorithm for minimum enclosing ball with outliers and extend it to the outlier recognition problem. By virtue of the "core-set" idea and the Random Gradient Descent Tree, we propose an efficient algorithm which is linear in the number of points n and the dimensionality d, and provides a probability bound. In experiments, compared with some existing outlier recognition algorithms, our method is proven to be efficient and robust to the outlier ratios.

In Chapter three, we adopt the "doubling dimension" to characterize the intrinsic dimension of a point set. By the property of doubling dimension, we can approximate the geometric alignment between two point sets by executing the existing alignment algorithms on their subsets, which achieves a much smaller time complexity. More importantly, the proposed approximate method has a theoretical upper bound and can serve as the preprocessing step of any alignment algorithm.

ACKNOWLEDGEMENTS

Firstly, I want to express my thanks to my parents and my brother for supporting me to go abroad and do research.

I also thank my advisor Prof. Hu Ding for leading me to the research on computational geometry and theoretical computer science.

Thanks to Prof. Eric Torng and Prof. Yiying Tong for being my thesis committee members.

I thank our department secretary Steven Smith for helping me a lot during my master stage.

Finally, I also would like to thank my fellows Liyang Xie, Kaixiang Lin, Qinhao Zhang, Jun Guo and my friends Qianqian, Zhengfa, Yue Wang and Shuo Wang.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ALGORITHMS

KEY TO ABBREVIATIONS

PCA principal component analysis

MDA multiple discriminant analysis

SVM support vector machine

MEB minimum enclosing ball

RGD random gradient descent

EMD earth mover's distance

PPI protein-protein interaction

PTAS polynomial-time approximation scheme

ICP iterative closest point

OP orthogonal procrustes

CHAPTER 1

INTRODUCTION

In the era of big data, we expect to replace some polynomial time algorithms with linear or even sub-linear time algorithms. Generally speaking, we focus on two aspects: *data quantity* and *data dimensionality*.

1. For the former case, intuitively we only employ a part of dataset by sampling some "important" data points. A relevant concept is called "core-set". Specifically, given a point set P and a class of functions $\mathcal{F}$, for any function $f \in \mathcal{F}$, we denote $\text{cost}(f, P)$ as the cost of function f on dataset P, then a (weighted) subset S of P is called the core-set of P if for $\epsilon > 0$ and any $f \in \mathcal{F}$ we have that

$$(1 - \epsilon) \cdot \text{cost}(f, P) \leq \text{cost}(f, S) \leq (1 + \epsilon) \cdot \text{cost}(f, P).$$

   For core-sets, we refer readers to several surveys (Phillips (2016); Clarkson (2010); Munteanu and Schwiegelshohn (2018)).

2. For the second case, the dimension reduction is the most commonly used technique in practice. There are a lot of classical methods in different fields including machine learning, statistics and theoretical computer science, such as principal component analysis (PCA), multiple discriminant analysis (MDA) and Johnson-Lindenstrauss lemma (Achlioptas (2003)). In practice, the method that we adopt is determined by the specific problems. For a survey, we refer readers to (Song et al. (2018)).

In this thesis, we concentrate on two machine learning problems in high dimension. Our motivation is the observation that many real-world datasets often have low intrinsic dimensions (Belkin (2003)). For example, the core-set size of minimum enclosing ball is $\lceil 1/\epsilon \rceil$ (Badoiu and Clarkson (2003)) which is independent of the number of points n and the dimensionality d; in computational geometry, the intrinsic dimension of a metric space is often characterized by the

concept “doubling dimension”, which is also independent of the dimensionality of the space. Based on the key observation, in this thesis we improve the time complexities of two important problems in machine learning. Specifically, the contributions of this thesis are summarized as follows.

1. We present a novel geometric approach for outlier recognition. An approximate minimum enclosing ball is built to cover the inliers but exclude the outliers. However, the existence of outliers makes the problem to be not only non-convex but also highly combinatorial, and the high dimensionality makes the problem more difficult. To tackle these challenges, we develop a randomized algorithmic framework using the core-set idea. Comparing with existing methods for outlier recognition, our method has a quality guarantee and lower time complexity.

2. We are the first to develop efficient algorithmic framework for the geometric alignment problem in large-scale and high dimension. By virtue of the property of doubling dimension, we prove that the given geometric patterns with low doubling dimension can be substantially compressed and thus save a lot running time when computing alignment. More importantly, our method is an independent step, therefore it can serve as the preprocessing for various alignment algorithms.

CHAPTER 2

A NOVEL GEOMETRIC APPROACH FOR OUTLIER RECOGNITION IN HIGH DIMENSION

2.1 Introduction

In this big data era, we are confronted with an extremely large amount of data and it is important to develop efficient algorithmic techniques to handle the arising realistic issues. Due to its recent rapid development, deep learning (Hinton et al. (2012)) becomes a powerful tool for many emerging applications; meanwhile, the quality of training dataset often plays a key role and seriously affects the final learning result. For example, we can collect tremendous data (e.g., texts or images) through the internet, however, the obtained dataset often contains a significant amount of outliers. Since manually removing outliers will be very costly, it is very necessary to develop some efficient algorithms for recognizing outliers automatically in many scenarios.

Outlier recognition is a typical unsupervised learning problem and its counterpart in supervised learning is usually called *anomaly detection* (Tan, Steinbach, and Kumar (2006)). In anomaly detection, the given training data are always positive and the task is to generate a model to depict the positive samples. Therefore, any new data can be distinguished to be positive or negative (i.e., anomaly) based on the obtained model. Several existing methods, especially for image data, include autoencoder (Sakurada and Yairi (2014)) and sparse coding (Lu, Shi, and Jia (2013)).

Unlike anomaly detection, the given data for outlier recognition are unlabeled; thus we can only model it as an optimization problem based on some reasonable assumption in practice. For instance, it is very natural to assume that the inliers (i.e., normal data) locate in some dense region while the outliers are scattered in the feature space. Actually, many well known outlier recognition methods are based on this assumption (Breunig et al. (2000); Ester et al. (1996)). However, most of the density-based methods are only for low-dimensional space and quite limited for large-scale high-dimensional data that are very common in computer vision problems (note that several high-

dimensional approaches often are of heuristic natures and need strong assumptions (Kriegel et al. (2009); Kriegel, Schubert, and Zimek (2008); Aggarwal and Yu (2001)). Recently, (Liu, Hua, and Smith (2014)) applied the one-class support vector machine (SVM) method (Schölkopf et al. (1999)) to high-dimensional outlier recognition. Further, (Xia et al. (2015)) introduced a new unsupervised model of autoencoder inspired by the observation that inliers usually have smaller reconstruction errors than outliers.

Although the aforementioned methods could efficiently solve the problem of outlier recognition to a certain extent, they still suffer from several issues such as high time and space complexities or unstable performances for different datasets. In this chapter, we present a novel geometric approach for outlier recognition. Roughly speaking, we try to build an approximate minimum enclosing ball (MEB) to cover the inliers but exclude the outliers. This model is seemed to be very simple but involves a couple of computational challenges. For example, the existence of outliers makes the problem to be not only non-convex but also highly combinatorial. Also, the high dimensionality makes the problem more difficult. To tackle these challenges, we develop a randomized algorithmic framework using a popular geometric concept called "core-set". Comparing with existing results for outlier recognition, we provide a thorough analysis on the complexities and quality guarantee. Finally, we test our algorithm on both random and benchmark datasets and the experimental results reveal the advantages of our approach over various existing methods.

2.1.1 Related Work

Besides the aforementioned existing results, many other methods for outlier recognition/anomaly detection were developed previously and the readers can refer to several excellent surveys (Kriegel, Kröger, and Zimek (2009); Chandola, Banerjee, and Kumar (2009); Gupta et al. (2014)).

In computational geometry, a core-set (Agarwal, Har-Peled, and Varadarajan (2005)) is a small set of points that approximates the shape of a much larger point set, and thus can be used to significantly reduce the time complexities for many optimization problems (please refer to a recent survey (Phillips (2016))). In particular, a core-set can be applied to efficiently compute an

approximate MEB for a set of points in high-dimensional space (Badoiu, Har-Peled, and Indyk (2002); Kumar, Mitchell, and Yildirim (2003)). Moreover, (Bădoiu and Clarkson (2008); Badoiu and Clarkson (2003)) showed that it is possible to find a core-set of size $\lceil 1/\epsilon \rceil$ that yields a $(1+\epsilon)$-approximate MEB, with an important advantage that the size is independent of the original size and dimensionality of the dataset. In fact, the algorithm for computing the core-set of MEB is a *Frank-Wolfe* style algorithm (Frank and Wolfe (1956)), which has been systematically studied by Clarkson (Clarkson (2010)).

The problem of MEB with outliers also falls under the umbrella of the topic *robust shape fitting* (Har-Peled and Wang (2004); Agarwal, Har-Peled, and Yu (2008)), but most of the approaches cannot be applied to high-dimensional data. (Zarrabi-Zadeh and Mukhopadhyay (2009)) studied MEB with outliers in high dimension, however, the resulting approximation is a constant 2 that is not fit enough for the applications proposed in this chapter.

Actually, our idea is inspired by a recent work about removing outliers for SVM (Ding and Xu (2015)), where they proposed a novel combinatorial approach called *Random Gradient Descent (RGD) Tree*. It is known that SVM is equivalent to finding the *polytope distance* from the origin to the *Minkowski Difference* of the given two labeled point sets. Gilbert algorithm (Gilbert (1966); Gärtner and Jaggi (2009)) is an efficient Frank-Wolfe algorithm for computing polytope distance, but a significant drawback is that the performance is too sensitive to outliers. To remedy this issue, RGD Tree accommodates the idea of randomization to Gilbert algorithm. Namely, it selects a small random sample in each step by a carefully designed strategy to overcome the adverse effect from outliers.

2.1.2 Preliminaries

As mentioned before, we model outlier recognition as a problem of MEB with outliers in high dimension. Here we first introduce several definitions that are used throughout this chapter.

Definition 1 (Minimum Enclosing Ball (MEB))**.** *Given a set P of points in $\mathbb{R}^d$, MEB is the ball covering all the points with the smallest radius. The MEB is denoted by $MEB(P)$.*

Definition 2 (MEB with Outliers)**.** *Given a set P of n points in $\mathbb{R}^d$ and a small parameter $\gamma \in (0, 1)$, MEB with outliers is to find the smallest ball that covers at least $(1-\gamma)n$ points. Namely, the task is to find a subset of P having at least $(1-\gamma)n$ points such that the resulting MEB is the smallest among all the possible choices; the induced ball is denoted by $MEB(P, \gamma)$.*

From Definition 2 we can see that the major challenge is to determine the subset of P which makes the problem a challenging combinatorial optimization. Therefore we relax our goal to its approximation as follows. For the sake of convenience, we always use P_{opt} to denote the optimal subset of P, that is, $P_{\text{opt}} = \arg_{P'} \min\{$ the radius of $MEB(P') \mid P' \subset P, |P'| \geq (1-\gamma)n\}$, and r_{opt} to denote the radius of $MEB(P_{\text{opt}})$.

Definition 3 (Bi-criteria Approximation)**.** *Given an instance (P, γ) for MEB with outliers and two small parameters $0 < \epsilon, \delta < 1$, an (ϵ, δ)-approximation is a ball that covers at least $(1-(1+\delta)\gamma)n$ points and has the radius at most $(1+\epsilon)r_{opt}$.*

When both ϵ and δ are small, the bi-criteria approximation is very close to the optimal solution with only a slight violation on the number of covering points and radius.

2.2 Our Algorithm and Analyses

In this section, we present our method in detail. For the sake of completeness, we first briefly introduce the core-set for MEB based on the idea of (Badoiu and Clarkson (2003)).

The algorithm is a simple iterative procedure with an elegant analysis: initially, it selects an arbitrary point and places it into a set S that is empty at the beginning; in each of the following $\lceil 2/\epsilon \rceil$ steps, the algorithm finds the farthest point to the center of $MEB(S)$ and adds it to S; finally, the center of $MEB(S)$ induces a $(1+\epsilon)$-approximation for MEB of the whole input point set. The selected $\lceil 2/\epsilon \rceil$ points are also called the core-set of MEB. To ensure that there is at least certain extent of improvement achieved in each iteration, (Badoiu and Clarkson (2003)) showed that the following two inequalities would hold if the algorithm always selects the farthest point to

the temporary center of $MEB(S)$:

$$r_{i+1} \geq (1+\epsilon)r_{\text{opt}} - L_i, \tag{2.1}$$

$$r_{i+1} \geq \sqrt{r_i^2 + L_i^2}, \tag{2.2}$$

where r_i and r_{i+1} are the radii of the i-th and the $(i+1)$-th iterations respectively, r_{opt} is the optimal radius of the MEB, and L_i is the shifting distance of the center of $MEB(S)$.

Algorithm 1 (ϵ, δ)-approximation Algorithm of Outlier Recognition Problem

Input: A point set P with n points in $\mathbb{R}^d$, the fraction of outliers $\gamma \in (0,1)$ and four parameters $0 < \epsilon, \delta, \mu < 1, h \in \mathbb{Z}^+$.

Output: A tree with each node whose attached point is a candidate for the (ϵ, δ)-approximation solutions.

1: Each node v in the tree is associated with a point (with a slight abuse of notation, we also use v to denote the point). Initially, randomly pick a point from P as root node r.

2: Starting with root, grow each node as follows:

(1) Let v be the current node.

(2) If the height of v is h, v becomes a leaf node. Otherwise, perform the following steps:

(a) Let $\mathcal{P}_v^r$ denote the set of points along the path from root r to node v, and c_v denote the center of $MEB(\mathcal{P}_v^r)$. We say that c_v is the attached point of v.

(b) Let $k = (1+\delta)\gamma n$. Compute the point set P_v containing the top k points which have the largest distances to c_v.

(c) Take a random sample S_v of size $\left(1 + \frac{1}{\delta}\right) \ln \frac{h}{\mu}$ from P_v, and let each point $v' \in S_v$ be a child node of v.

2.2.1 Algorithm for MEB with Outliers

We present our (ϵ, δ)-approximation algorithm for MEB with outliers in this section. Although the outlier recognition problem belongs to unsupervised learning, we can estimate the fraction of outliers in the given data before executing our algorithm. In practice, we can randomly collect a small set of samples from the given data, and manually identify the outliers and estimate the outlier ratio γ. Therefore, in this chapter we assume that the outlier ratio is known.

To better understand our algorithm, we first illustrate the high-level idea. If taking a more careful analysis on the previously mentioned core-set construction algorithm (Badoiu and Clarkson

(2003)), we can find that it is not necessary to select the farthest point to the center of $MEB(S)$ in each step. Instead, as long as the selected point has a distance larger than $(1+\epsilon)r_{\text{opt}}$, the minimal extent of improvement would always be guaranteed (Ding and Xu (2011)). As a consequence, we investigate the following approach.

We denote the ball centered at point c with radius $r > 0$ as $Ball(c, r)$. Recall that P_{opt} is the subset of P yielding the optimal MEB with outliers, and r_{opt} is the radius of $MEB(P_{\text{opt}})$ (see Section 2.1.2). In the i-th step, we add an arbitrary point from $P_{\text{opt}} \setminus Ball(c_i, (1+\epsilon)r_{\text{opt}})$ to S where c_i is the current center of S. Based on the above observation, we know that a $(1+\epsilon)$-approximation is obtained after at most $\lceil 2/\epsilon \rceil$ steps, that is, $\left|P \cap Ball(c_i, (1+\epsilon)r_{\text{opt}})\right| \geq (1-\gamma)n$ when $i \geq \lceil 2/\epsilon \rceil$.

However, in order to carry out the above approach we need to solve two key issues: how to determine the value of r_{opt} and how to select a point belonging to $P_{\text{opt}} \setminus Ball(c_i, (1+\epsilon)r_{\text{opt}})$. Actually, we can implicitly avoid the first issue via replacing the radius $(1+\epsilon)r_{\text{opt}}$ by the k-th largest distance from the points of P to c_i, where k is some appropriate number that will be determined in our following analysis. For the second issue, we have to take a small random sample instead of a single point from $P_{\text{opt}} \setminus Ball(c_i, (1+\epsilon)r_{\text{opt}})$ and try each of the sampled points; this operation will result in a tree structure that is similar to the RGD Tree introduced by (Ding and Xu (2015)) for SVM. We present the algorithm in Algorithm 1 and place the detailed parameter settings, proof of correctness, and complexity analyses in Sections 2.2.2 & 2.2.3.

We illustrate Step 2(2)(a-c) of Algorithm 1 in Figure 2.1.

2.2.2 Parameter Settings and Quality Guarantee

We denote the tree constructed by Algorithm 1 as $\mathbb{H}$. The following theorem shows the success probability of Algorithm 1.

Theorem 1. *If we set $h = \lceil \frac{2}{\epsilon} \rceil + 1$, then with probability at least $(1-\mu)(1-\gamma)$ there exists at least one node of $\mathbb{H}$ yielding an (ϵ, δ)-approximation for the problem of MEB with outliers.*

Before proving Theorem 1, we need to introduce several important lemmas.

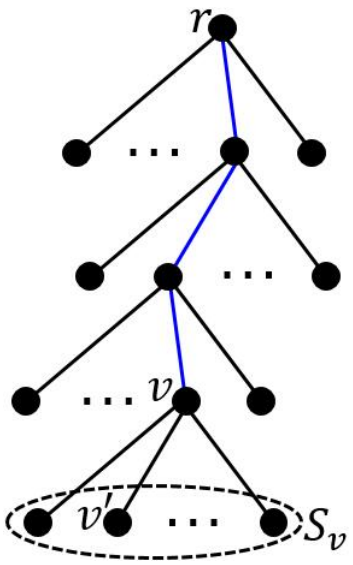


Figure 2.1: The blue links represent the path from root r to node v, and $\mathcal{P}_v^r$ contains the four points along the path. The point set S_v corresponds to the child nodes of v.

Lemma 1 (Ding and Xu (2014))**.** *Let Q be a set of elements, and Q' be a subset of Q with size $|Q'| = \beta |Q|$ for some $\beta \in (0, 1)$. If one randomly samples $\frac{1}{\beta} \ln \frac{1}{\eta}$ elements from Q, then with probability at least $1 - \eta$, the sample contains at least one element in Q' for any $0 < \eta < 1$.*

Lemma 2. *For each node v, the set S_v in Algorithm 1 contains at least one point from P_{opt} with probability $1 - \frac{\mu}{h}$.*

Proof. Since $|P_v| = (1 + \delta)n\gamma$ and $\left|P \backslash P_{\text{opt}}\right| = n\gamma$, we have

$$\begin{aligned} \frac{\left|P_v \cap P_{\text{opt}}\right|}{|P_v|} &= 1 - \frac{\left|P_v \backslash P_{\text{opt}}\right|}{|P_v|} \\ &\geq 1 - \frac{\left|P \backslash P_{\text{opt}}\right|}{|P_v|} \\ &= \frac{\delta}{1 + \delta}. \end{aligned} \tag{2.3}$$

Note that the size of S_v is $(1 + \frac{1}{\delta}) \ln \frac{h}{\mu}$. If we apply Lemma 1 via setting $\beta = \frac{\delta}{1+\delta}$ and $\eta = \frac{\mu}{h}$, it is easy to know that S_v contains at least one point from P_{opt} with probability $1 - \frac{\mu}{h}$. □

Lemma 3. *With probability $(1-\gamma)(1-\mu)$, there exists a leaf node $u \in \mathbb{H}$ such that the corresponding set $\mathcal{P}_u^r \subset P_{opt}$.*

Proof. Lemma 2 indicates that each node v has a child node corresponding to a point from P_{opt} with probability $1 - \frac{\mu}{h}$. In addition, the probability of root r belonging to P_{opt} is $1 - \gamma$ (recall that γ is the fraction of outliers). Note that the height of $\mathbb{H}$ is h, then with probability at least

$$(1 - \gamma)\left(1 - \frac{\mu}{h}\right)^h > (1 - \gamma)(1 - \mu), \tag{2.4}$$

there exists one leaf node $u \in \mathbb{H}$ satisfying $\mathcal{P}_u^r \subset P_{\text{opt}}$. □

In the remaining analyses, we always assume that such a root-to-leaf path $\mathcal{P}_u^r$ described in Lemma 3 exists and only focus on the nodes along this path. We denote $\hat{R} = (1 + \epsilon)r_{\text{opt}}$ where r_{opt} is the optimal radius of the MEB with outliers. Let $Ball(c_v, r_v)$ be the MEB covering $P \setminus P_v$ centered at c_v, and the radii of $MEB(\mathcal{P}_v^r)$ and $MEB(\mathcal{P}_{v'}^r)$ be $\tilde{r}_v$ and $\tilde{r}_{v'}$ respectively. Readers can refer to Figure 2.2. The following lemma is a key observation for MEB.

Lemma 4 (Badoiu and Clarkson (2003))**.** *Given a set P of points in $\mathbb{R}^d$, let r_P and c_P be the radius and center of $MEB(P)$ respectively. Then for any point $p \in \mathbb{R}^d$ with a distance $K \geq 0$ to c_P, there exists a point $q \in P$ such that $\|p - q\| \geq \sqrt{r_P^2 + K^2}$.*

The following lemma is a key for proving the quality guarantee of Algorithm 1. As mentioned in Section 2.2.1, the main idea follows the previous works (Badoiu and Clarkson (2003); Ding and Xu (2011)). For the sake of completeness, we present the detailed proof here.

Lemma 5. *For each node $v \in \mathcal{P}_u^r$, at least one of the following two events happens: (1) c_v is an (ϵ, δ)-approximation; (2) its child v' on the path $\mathcal{P}_u^r$ satisfies*

$$\tilde{r}_{v'} \geq \frac{\hat{R}}{2} + \frac{\tilde{r}_v^2}{2\hat{R}}. \tag{2.5}$$

Proof. If $r_v \leq \hat{R}$, then we are done; that is, $Ball(c_v, r_v)$ covers $(1 - (1 + \delta)\gamma)n$ points and $r_v \leq (1 + \epsilon)r_{\text{opt}}$. Otherwise, $r_v > \hat{R}$ and we consider the second case.

By triangle inequality and the fact that v' (i.e., the point associating the node "v'") lies outside $Ball(c_v, r_v)$, we have

$$\left\|c_v - c_{v'}\right\| + \left\|c_{v'} - v'\right\| \geq \left\|c_v - v'\right\| > r_v > \hat{R}. \tag{2.6}$$

Let $\|c_v - c_{v'}\| = K_v$. Combining the fact that $\|c_{v'} - v'\| \leq \tilde{r}_{v'}$, we have

$$\tilde{r}_{v'} > \hat{R} - K_v. \tag{2.7}$$

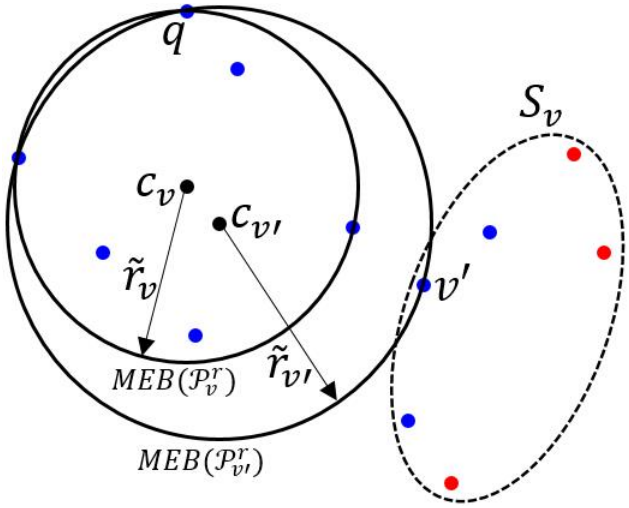


Figure 2.2: The illustration of $MEB(\mathcal{P}_v^r)$ and $MEB(\mathcal{P}_{v'}^r)$; the blue and red points represent the inliers and outliers, respectively.

By Lemma 4, we know that there exists one point q (see Figure 2.2) in $MEB(\mathcal{P}_v^r)$ satisfying $\|q - c_{v'}\| \geq \sqrt{\tilde{r}_v^2 + K_v^2}$. Since q is also inside $MEB(\mathcal{P}_{v'}^r)$, $\|q - c_{v'}\| \leq \tilde{r}_{v'}$. Then, we have

$$\tilde{r}_{v'} \geq \sqrt{\tilde{r}_v^2 + K_v^2}. \tag{2.8}$$

Combining (2.7) and (2.8), we obtain

$$\tilde{r}_{v'} \geq \max\left\{\hat{R} - K_v, \sqrt{\tilde{r}_v^2 + K_v^2}\right\}. \tag{2.9}$$

Because $\hat{R} - K_v$ and $\sqrt{\tilde{r}_v^2 + K_v^2}$ are decreasing and increasing on K_v respectively, we let $\hat{R} - K_v = \sqrt{\tilde{r}_v^2 + K_v^2}$ to achieve the lower bound (i.e., $K_v = \frac{\hat{R}^2 - \tilde{r}_v^2}{2\hat{R}}$). Substituting the value of K_v to (2.9), we have $\tilde{r}_{v'} \geq \frac{\hat{R}}{2} + \frac{\tilde{r}_v^2}{2\hat{R}}$. As a consequence, the second event happens and the proof is completed. □

Now we prove Theorem 1 by the idea from (Badoiu and Clarkson (2003)). Suppose no node in $\mathcal{P}_u^r$ makes the first event of Lemma 5 occur. As a consequence, we obtain a series of inequalities

for each pair of radii $\tilde{r}_{v'}$ and $\tilde{r}_v$ (see (2.5)). For ease of analysis, we denote $\tilde{r}_v = \lambda_i \hat{R}$ if the height of v is i in $\mathbb{H}$. By Inequality (2.5), we have

$$\lambda_{i+1} \geq \frac{1 + \lambda_i^2}{2}. \tag{2.10}$$

Combining the initial case $\lambda_1 = 0$ and (2.10), we obtain

$$\lambda_i \geq 1 - \frac{2}{i+1} \tag{2.11}$$

by induction (Badoiu and Clarkson (2003)). Note that the equality in (2.11) holds only when $i = 1$, therefore,

$$\lambda_h > 1 - \frac{2}{h+1} = 1 - \frac{2}{\lceil \frac{2}{\epsilon} \rceil + 2} \geq 1 - \frac{2}{\frac{2}{\epsilon} + 2} = \frac{1}{1+\epsilon}. \tag{2.12}$$

Then, $\tilde{r}_u = \lambda_h \hat{R} > r_{\text{opt}}$ (recall that u is the leaf node on the path $\mathcal{P}_u^r$), which is a contradiction to our assumption $\mathcal{P}_u^r \subset P_{\text{opt}}$. The success probability directly comes from Lemma 3. Overall, we obtain Theorem 1.

2.2.3 Complexity Analyses

We analyze the time and space complexities of Algorithm 1 in this section.

Time Complexity. For each node v, we need to compute the corresponding $MEB(\mathcal{P}_v^r)$. To avoid computing the exact MEB costly, we apply the approximation algorithm proposed by (Badoiu and Clarkson (2003)). See Algorithm 2 for details.

Algorithm 2 Approximation Algorithm of MEB

Input: A point set Q in $\mathbb{R}^d$, and $N \in Z^+$.

1: Start with an arbitrary point $c_1 \in Q$, $t \leftarrow 1$.
2: **while** $t < N$ **do**
3: Find the point $q \in Q$ farthest away from c_t.
4: $c_{t+1} \leftarrow c_t + \frac{1}{t+1}(q - c_t)$.
5: $t \leftarrow t + 1$.
6: **end while**
7: **return** c_t.

For Algorithm 2, we have the following theorem.

Theorem 2 (Badoiu and Clarkson (2003))**.** *Let the center and radius of $MEB(Q)$ be c_Q and r_Q respectively, then $\forall t$, $\left\|c_Q - c_t\right\| \leq \frac{r_Q}{\sqrt{t}}$.*

From Theorem 2, we know that a $(1+\varepsilon)$-approximation for MEB can be obtained when $N = 1/\varepsilon^2$ with the time complexity $O\left(\frac{|Q|d}{\varepsilon^2}\right)$. Suppose the height of node v is i, then the complexity for computing the corresponding approximate $MEB(\mathcal{P}_v^r)$ is $O\left(\frac{id}{\varepsilon^2}\right)$. Further, in order to obtain the point set P_v, we need to find the pivot point that has the $(n-k)$-th smallest distance to c_v. Here we apply the PICK algorithm (Blum et al. (1973)) which can find the l-th smallest from a set of n ($l \leq n$) numbers in linear time. Consequently, the complexity for each node v at the i-th layer is $O\left(\left(n + \frac{i}{\varepsilon^2}\right) d\right)$. Recall that there are $|S_v|^{i-1}$ nodes at the i-th layer of $\mathbb{H}$. In total, the time complexity of our algorithm is

$$T = \sum_{i=1}^{h} \left(\left(1 + \frac{1}{\delta}\right) \ln \frac{h}{\mu}\right)^{i-1} \left(n + \frac{i}{\varepsilon^2}\right) d. \tag{2.13}$$

If we assume $1/\varepsilon$ is a constant, the complexity $T = O(nd)$ is linear in n and d, where the hidden constant $C = \left(\left(1 + \frac{1}{\delta}\right) \ln \frac{h}{\mu}\right)^{h-1}$. In our experiments, we can carefully choose the parameters δ, ϵ, μ so as to keep the value of C not too large.

Space Complexity. In our implementation, we use a queue Q to store the nodes in the tree. When the head of Q is popped, its $|S_v|$ child nodes are pushed into Q. In other words, we simulate *breadth first search* on the tree $\mathbb{H}$. Therefore, Q always keeps its size at most $C = \left(\left(1 + \frac{1}{\delta}\right) \ln \frac{h}{\mu}\right)^{h-1}$. Note that each node v needs to store $\mathcal{P}_v^r$ to compute its corresponding MEB, but actually we only need to record the pointers to link the points in $\mathcal{P}_v^r$. Therefore, the space complexity of Q is $O(Ch)$. Together with the space complexity of the input data, the total space complexity of our algorithm is $O(Ch + nd)$.

2.2.4 Boosting

By Theorem 1, we know that with probability at least $(1-\mu)(1-\gamma)$ there exists an (ϵ, δ)-approximation in the resulting tree. However, when outlier ratio is high, say $\gamma = 0.5$, the success

probability $(1-\gamma)(1-\mu)$ will become small. To further improve the performance of our algorithm, we introduce the following two boosting methods.

1. **Constructing a forest.** Instead of building a single tree, we randomly initialize several root nodes and grow each root node to be a tree. Suppose the number of root nodes is κ. The probability that there exists an (ϵ, δ)-approximation in the forest is at least $1-(1-(1-\gamma)(1-\mu))^{\kappa}$ which is much larger than $(1-\gamma)(1-\mu)$.

2. **Sequentialization.** First, initialize one root node and build a tree. Then select the best node in the tree and set it to be the root node for the next tree. After iteratively performing the procedure for several rounds, we can obtain a much more robust solution.

2.3 Experiments

From our analysis in Section 2.2.2, we know that Algorithm 1 results in a tree $\mathbb{H}$ where each node v has a candidate c_v for the desired (ϵ, δ)-approximation for the problem of MEB with outliers. For each candidate, we identify the nearest $(1-(1+\delta)\gamma)n$ points to c_v as the inliers. To determine the final solution, we select the candidate that has the smallest variance of the inliers.

2.3.1 Datasets and Methods to Be Compared

In our experiments, we test the algorithms on two random datasets and two benchmark image datasets. In terms of the random datasets, we generate the data points based on normal and uniform distributions under the assumption that the inliers usually locate in dense regions while the outliers are scattered in the space. The benchmark image datasets include the popular MNIST (LeCun et al. (1998)) and Caltech (Fei-Fei, Fergus, and Perona (2007)). All of the experiments are performed on Windows workstation with 2.4GHz Intel Xeon CPU and 32GB DDR4 Memory.

To make our experiment more convincing, we compare our algorithm with three well known methods for outlier recognition: angle-based outlier detection (ABOD) (Kriegel, Schubert, and Zimek (2008)), one-class SVM (OCSVM) (Schölkopf et al. (1999)), and discriminative recon-

structions in an autoencoder (DRAE) (Xia et al. (2015)). Specifically, ABOD distinguishes the inliers and outliers by assessing the distribution of the angles determined by each 3-tuple data points; OCSVM models the problem of outlier recognition as a soft-margin one-class SVM; DRAE applies autoencoder to separate the inliers and outliers based on their reconstruction errors.

The performances of the algorithms are measured by the commonly used metric $F1$ score $= \frac{2*\text{Precision}*\text{Recall}}{\text{Precision}+\text{Recall}}$, where precision is the proportion of the correctly identified positives relative to the total number of identified positives, and recall is the proportion of the correctly identified positives relative to the total number of positives in the dataset.

2.3.2 Random Datasets

We validate our algorithm on the following two random datasets.

A toy example in 2D. To better illustrate the intuition of our algorithm, we first run it on a random dataset in 2D. We generate an instance of $10,000$ points with the outlier ratio $\gamma = 0.4$. The inliers are generated by a normal distribution; the outliers consist of four groups where the first three are generated by normal distributions and the last is generated by a uniform distribution. The four groups of outliers contains 800, 1200, 800, and 1200 points, respectively. See Figure 2.3. The red circle obtained by our algorithm is the boundary to distinguish the inliers and outliers where the resulting $F1$ score is 0.944. From this case, we can see that our algorithm can efficiently recognize the densest region even if the outlier ratio is high and the outliers also form some dense regions in the space.

High-Dimensional Points. We further test our algorithm and the other three methods on high-dimensional dataset. Similar to the previous 2D case, we generate $20,000$ points with four groups of outliers in $\mathbb{R}^{100}$; the outlier ratio γ varies from 0.1 to 0.5.

The $F1$ scores are displayed in Table 2.1, from which we can see that our algorithm significantly outperforms the other three methods for all the levels of outlier ratio.

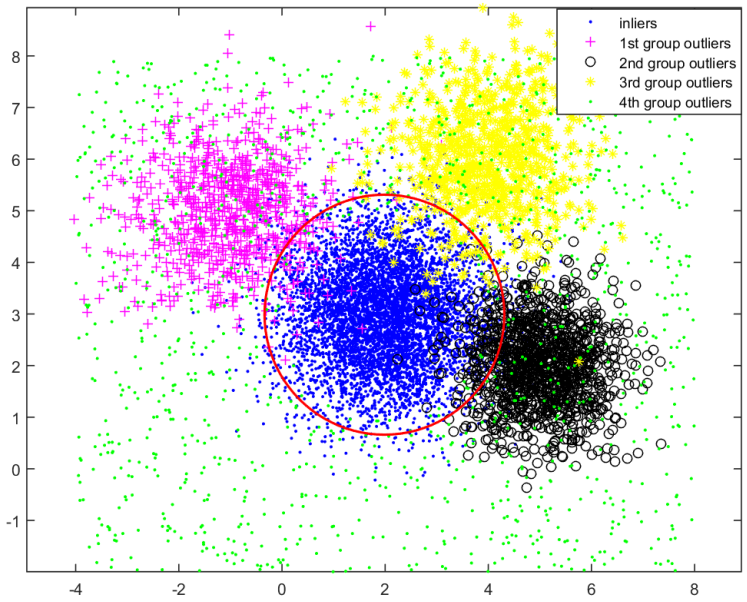


Figure 2.3: The illustration of our algorithm on a 2-dimensional point set.

Table 2.1: The $F1$ scores for the high-dimensional random dataset.

γ \ Algs	ABOD	OCSVM	DRAE	Ours
0.1	0.907	0.967	0.951	**0.984**
0.2	0.815	0.926	0.889	**0.965**
0.3	0.705	0.880	0.809	**0.939**
0.4	0.586	0.827	0.709	**0.938**
0.5	0.419	0.745	0.572	**0.898**

2.3.3 Benchmark Image Datasets

In this section, we evaluate all the four methods on two benchmark image datasets.

1. **MNIST Dataset**

   MNIST contains $70,000$ handwritten digits (0 to 9) composed of both training and test datasets. For each of the 10 digits, we add the outliers by randomly selecting the images from the other 9 digits. For each outlier ratio γ, we compute the average $F1$ score over all the 10 digits. To map the images to a feature (Euclidean) space, we use two kinds of image

features: PCA-grayscale and autoencoder feature.

(1) **PCA-grayscale Feature.** Each image in MNIST has a 28×28 grayscale which is represented by a 784-dimensional vector. Note that the images of MNIST have massive redundancy. For example, the digits often locate in the middle of the images and all the background pixels have the value 0. Therefore, we apply principal component analysis (PCA) to reduce the redundancy by trying multiple projection matrices which preserve 95%, 50%, and 10% energy of the original grayscale features. These three features are denoted as PCA-0.95, PCA-0.5 and PCA-0.1, respectively. The results are shown in Table 2.2. We notice that our $F1$ scores always achieve the highest by PCA-0.5; this is due to the fact that PCA-0.5 can significantly reduce the redundancy as well as preserve the most useful information (comparing with PCA-0.95 and PCA-0.1).

Table 2.2: The $F1$ scores of the four methods on MNIST by using PCA-grayscale; the three columns for each γ correspond to PCA-0.95, PCA-0.5 and PCA-0.1, respectively.

γ \ Algs	ABOD	OCSVM	DRAE	Ours
0.1	0.898, 0.895, 0.892	0.937, **0.941**, 0.934	0.913, 0.908, 0.911	**0.939**, **0.941**, **0.936**
0.2	0.775, 0.774, 0.771	0.874, 0.883, 0.867	0.822, 0.818, 0.816	**0.881**, **0.891**, **0.880**
0.3	0.648, 0.617, 0.642	0.804, 0.817, 0.798	0.726, 0.722, 0.711	**0.822**, **0.853**, **0.823**
0.4	0.500, 0.470, 0.496	0.725, 0.740, 0.713	0.620, 0.617, 0.602	**0.760**, **0.778**, **0.773**
0.5	0.346, 0.329, 0.364	**0.648**, 0.639, 0.605	0.531, 0.501, 0.488	0.633, **0.658**, **0.651**

(2) **Autoencoder Feature.** Autoencoder (Rumelhart, Hinton, and Williams (1988)) is often adopted to extract the features of grayscale images. The autoencoder model trained in our experiment has seven symmetrical hidden layers (1000-500-250-60-250-500-1000), and the input layer is a 784-dimensional grayscale. We use the features output by the middle hidden layer. The results are shown in Table 2.3 and our method achieves the best for most of the cases.

2. **Caltech Dataset**

The Caltech-256 dataset[1] includes 256 image sets. We choose 11 concepts as the inliers in

[1] `http://www.vision.caltech.edu/Image_Datasets/Caltech256/`

Table 2.3: The $F1$ scores of the four methods on MNIST by using autoencoder feature.

γ \ Algs	ABOD	OCSVM	DRAE	Ours
0.1	0.894	0.906	**0.933**	0.932
0.2	0.778	0.807	0.883	**0.885**
0.3	0.637	0.706	0.819	**0.831**
0.4	0.479	0.598	0.737	**0.770**
0.5	0.313	0.496	0.625	**0.694**

our experiment, which are *airplane, binocular, bonsai, cup, face, ketch, laptop, motorbike, sneaker, t-shirt*, and *watch*. We apply VGG net (Simonyan and Zisserman (2014)) to extract the image features, which have 4096 dimensions output by the second fully-connected layer. The results are shown in Table 2.4.

Table 2.4: The $F1$ scores of the four methods on Caltech-256 by using VGG net feature.

γ \ Algs	ABOD	OCSVM	DRAE	Ours
0.1	0.945	0.930	0.955	**0.964**
0.2	0.838	0.885	0.937	**0.948**
0.3	0.707	0.839	0.930	**0.932**
0.4	0.499	0.783	**0.927**	0.924
0.5	0.233	0.739	**0.912**	0.906

Unlike the random data, the distribution of real data in the space is much more complicated. To alleviate this problem, we try to capture the key parts of the original VGG net feature. Similar to MNIST dataset, we apply PCA to reduce the redundancy of VGG net feature. Three matrices are obtained to preserve 95%, 50%, and 10% energy respectively. The results are shown in Table 2.5, from which we can see that our method achieves the best for all the cases, especially when using PCA-0.5 (marked by underlines). More importantly, PCA-0.5 considerably improves the results that use the original VGG net features (compare Table 2.5 with Table 2.4), and has only 50 dimensions, which significantly reduces the running time.

Table 2.5: The $F1$ scores of the four methods on Caltech-256 by using PCA-VGG feature; the three columns for each γ correspond to PCA-0.95, PCA-0.5 and PCA-0.1, respectively.

γ \ Algs	ABOD	OCSVM	DRAE	Ours
0.1	0.944, 0.942, 0.941	0.932, 0.914, 0.921	0.955, 0.947, 0.928	**0.966**, **0.986**, **0.949**
0.2	0.837, 0.832, 0.869	0.884, 0.894, 0.867	0.918, 0.924, 0.878	**0.950**, **0.984**, **0.923**
0.3	0.707, 0.708, 0.715	0.837, 0.869, 0.827	0.873, 0.914, 0.835	**0.934**, **0.978**, **0.897**
0.4	0.497, 0.489, 0.525	0.782, 0.830, 0.771	0.873, 0.902, 0.773	**0.916**, **0.973**, **0.871**
0.5	0.223, 0.199, 0.288	0.717, 0.790, 0.699	0.869, 0.887, 0.692	**0.899**, **0.958**, **0.844**

2.3.4 Comparisons of Time Complexities

From Section 2.3.2 and Section 2.3.3 we know that our method achieves the robust and competitive performances in terms of accuracy. In this section, we compare the time complexities of all the four algorithms.

ABOD has the time complexity $O(n^3d)$. In the experiments, we use its speed-up edition *FastABOD* which has the reduced time complexity $O((n^2 + nk^2)d)$ where k is some specified parameter.

OCSVM is formulated as a quadratic programming with the time complexity $O(n^3)$.

DRAE alternatively executes the following two steps: discriminative labeling and reconstruction learning. Suppose it runs N_1 rounds; actually the two inner steps are also iterative procedures which both run N_2 iterations. Thus, the total time complexity of DRAE is $O(N_1N_2hdn)$, where h is the number of the hidden layer nodes that can be generally expressed as d/m (m is a constant); then the total time complexity becomes $O(nd^2)$ where the hidden constant $\tilde{C}$ is a large constant depending on N_1, N_2 and m.

When the number of points n is large, FastABOD, OCSVM, and DRAE will be very time-consuming. On the contrary, our algorithm takes only linear running time (see Section 2.2.3) and usually runs much faster in practice.

2.4 Summary

In this chapter, we present a new approach for outlier recognition in high dimension. Most existing methods have high time and space complexities or cannot achieve a quality guaranteed

solution. On the contrary, our method yields a nearly optimal solution with the time and space complexities linear in the input size and dimensionality. Furthermore, the experimental results indicate that our approach outperforms several popular methods in terms of accuracy.

CHAPTER 3

GEOMETRIC ALIGNMENT IN LOW DOUBLING DIMENSION

3.1 Introduction

Given two geometric patterns, the problem of alignment is to find their appropriate spatial positions so as to minimize their difference. In general, a geometric pattern is represented by a set of (weighted) points in a space, and their difference is often measured by some objective function. Figure 3.1 presents an illustration of alignment.

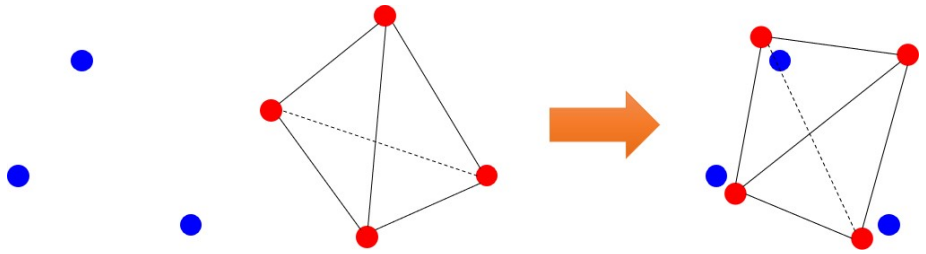


Figure 3.1: The illustration of alignment of two geometric patterns.

In this chapter, we focus on the geometric alignment. Before introducing the definition of geometric alignment, we first define Earth Mover's Distance (EMD) and rigid transformation.

Definition 4 (EMD). *Let $A = \{a_1, \cdots, a_{n_1}\}$ and $B = \{b_1, \cdots, b_{n_2}\}$ be two sets of weighted points in $\mathbb{R}^d$ with non-negative weights α_i and β_j for each $a_i \in A$ and $b_j \in B$ respectively, and $W_A = \sum_{i=1}^{n_1} \alpha_i, W_B = \sum_{j=1}^{n_2} \beta_j$ be their respective total weights. The EMD between A and B is defined as*

$$\mathcal{EMD}(A, B) = \frac{1}{\min\{W_A, W_B\}} \min_{F} \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} f_{ij} \left\| a_i - b_j \right\|^2, \tag{3.1}$$

where $F = \{f_{ij}, i = 1, \cdots, n_1, j = 1, \cdots, n_2\}$ is a feasible flow from A to B, i.e., $f_{ij} \geq 0$, $\sum_{i=1}^{n_1} f_{ij} \leq \beta_j$, $\sum_{j=1}^{n_2} f_{ij} \leq \alpha_i$, and $\sum_{i=1}^{n_1} \sum_{j=1}^{n_2} f_{ij} = \min\{W_A, W_B\}$.

See Figure 3.2 for an illustration of EMD. Intuitively, EMD can be viewed as the minimum transportation cost between A and B, where the weights of A and B are the "supplies" and "demans"

respectively, and the cost of each edge connecting a pair of points from A to B is their "ground distance". Generally, the "ground distance" has various forms, and here we simply use the Euclidean distance. In addition, the major advantage of EMD over other measures is its robustness with regard to noise in practice.

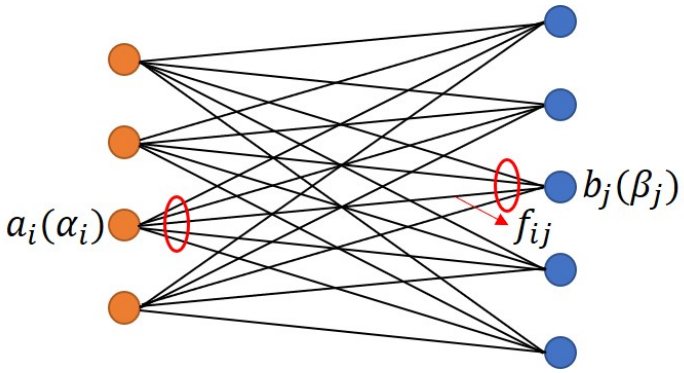


Figure 3.2: The illustration of EMD between two weighted point sets.

Definition 5 (Rigid Transformation)**.** *Let P be a set of points in $\mathbb{R}^d$. A rigid transformation $\mathcal{T}$ on P is a transformation (i.e., rotation, translation, reflection, or their combination) which preserves the pairwise distances of the points in P.*

We consider rigid transformation for alignment, because it is very natural to interpret in real-world problems. Given the definitions of EMD and rigid transformation, here we formally define geometric alignment.

Definition 6 (Geometric Alignment)**.** *Given two weighted point sets A and B as described in Definition 4, the geometric alignment between A and B under rigid transformation is to determine a rigid transformation $\mathcal{T}$ for B so as to minimize the EMD $\mathcal{EMD}(A, \mathcal{T}(B))$.*

In practice, geometric alignment has many applications in the field of computer vision such as image retrieval and matching (Cohen and Guibas (1999)), fingerprint recognition (Maltoni et al. (2009)) and face alignment (Cao et al. (2014)). Besides the computer vision applications in 2D or 3D, there are a lot of high dimensional problems that can be solved by geometric alignment. Here we briefly introduce several examples.

(1) Natural language processing

Some research on natural language processing has indicated that different languages often share some similar structure at the word level (Youn et al. (2016)); particularly, recent study on word semantics embedding has also shown that there exists structural isomorphism across languages (Mikolov, Le, and Sutskever (2013)), and EMD is a good distance metric for languages and documents (Zhang et al. (2017); Kusner et al. (2015)). Therefore, (Zhang et al. (2017)) proposed to learn the transformation between different languages without any cross-lingual supervision, and the problem is reduced to minimizing the EMD by finding the optimal geometric alignment in high dimension. Figure 3.3 shows us an illustration of geometric alignment of word embeddings.

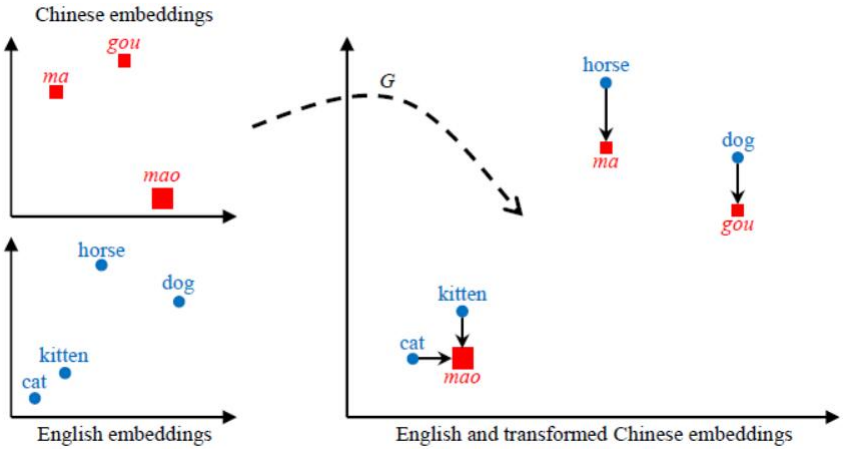


Figure 3.3: This image is excerpted from (Zhang et al. (2017)). After obtaining Chinese and English word embeddings, the problem becomes finding the optimal geometric alignment to minimize their EMD.

(2) Protein-Protein interaction (PPI) network

A PPI network is a graph representing the interactions among proteins. Given two PPI networks, it is a fundamental bioinformatics problem to find their alignment in order to understand the correspondences between different species (Malod-Dognin, Ban, and Pržulj (2017)). However, most existing methods need to solve the subgraph isomorphism problem which is NP-hard, and often suffer from a high computational complexity. To circumvent this issue, (Liu et al. (2017))

developed a new framework based on geometric alignment in Euclidean space. See Figure 3.4 for an illustration of PPI networks alignment.

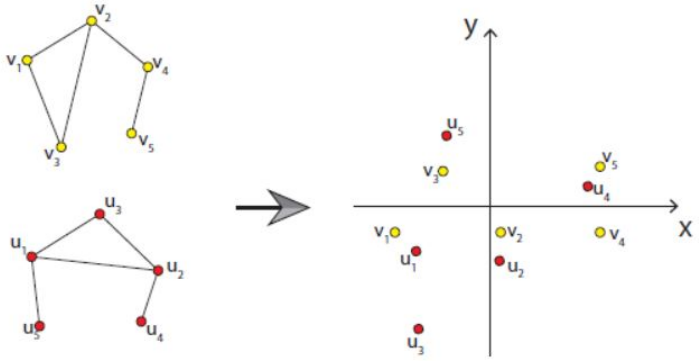


Figure 3.4: The illustration of PPI networks alignment (Liu et al. (2017)).

Despite of a number of applications in reality, the research on geometric alignment algorithms is quite limited and far from being satisfactory. Basically, we need to take into account high dimensionality and large number of points of the geometric patterns simultaneously. Unfortunately, even for the 2D and 3D cases, it is quite challenging to solve the geometric alignment problem.

In this chapter, we develop an efficient algorithmic framework for the geometric alignment problem with large size and high dimensionality. Our key observation is that many real-world datasets often manifest low intrinsic dimensions (Belkin (2003)). For example, human handwriting images can be well embedded into some low dimensional manifold though their Euclidean dimensions can be very high (Tenenbaum, De Silva, and Langford (2000)). Inspired by the observation, we adopt the widely used concept "doubling dimension" (Krauthgamer and Lee (2004); Talwar (2004); Karger and Ruhl (2002); Har-Peled and Mendel (2006); Dasgupta and Sinha (2013)) to characterize the intrinsic dimension in geometric alignment problems. We prove that when computing geometric alignment, the geometric patterns with low doubling dimensions can be substantially compressed and thus save a lot running time. What's more, our proposed compression method is an independent algorithm which can be utilized for various alignment approaches.

3.1.1 Related Work

In this section, we recall some related work.

Given a bipartite graph where the two sets of nodes correspond to the points of A and B respectively and each edge connecting a_i and b_j has the weight $\left\|a_i - b_j\right\|^2$, then computing $\mathcal{EMD}(A, B)$ is actually a min-cost flow problem. A lot of min-cost flow algorithms have been proposed in the past decades, such as *network simplex algorithm* (Magnanti and Orlin (1993)) which is a specialized version of the simplex algorithm. Computing EMD is an instance of min-cost flow problem in Euclidean space, and a lot of faster algorithms have been proposed in computational geometry (Indyk (2007); Cabello et al. (2008); Agarwal et al. (2019)). However, most of these methods only work in low dimensional cases.

Actually, it is more challenging to find the geometric alignment between two sets because it requires one rigid transformation and EMD flows simultaneously. Moreover, because of the variety of rigid transformations, we can not employ the EMD embedding techniques (Indyk and Thaper (2003); Andoni et al. (2009)) to alleviate these issues. Specifically, the embedding can only preserve the EMD between A and B, however, since we do not know the rigid transformation $\mathcal{T}$ in advance, there are infinite $\mathcal{T}$ for set B, therefore, it is difficult to preserve the EMD between A and $\mathcal{T}(B)$. In addition, some theoretical methods have been proposed. (Klein and Veltkamp (2005)) presented an $O\left(2^{d-1}\right)$-approximation algorithm in d-dimensional, then (Cabello et al. (2008)) proposed a $(2 + \epsilon)$-approximation solution to the 2D case. Recently, (Ding and Xu (2017)) developed a polynomial-time approximation scheme (PTAS) for constant dimensionality. Unfortunately, these theoretical methods are impractical when dimensionality is not constant. (Ding and Xu (2017)) also proved that any constant factor approximation has a time complexity at least $n^{\Omega(d)}$ based on some reasonable assumptions in the computational complexity theory, where $n = \max\{|A|, |B|\}$, i.e., it is unlikely to obtain a $(1 + \epsilon)$-approximation within practical running time, especially when n is large.

3.1.2 Outline of Our Work

In this section, we briefly introduce our work.

Our work is mainly based on (Cohen and Guibas (1999)), which proposed an alternating minimization method to obtain the geometric alignment between two point sets. Roughly speaking, this method is similar to the *Iterative Closest Point* (ICP) method (Besl and McKay (1992)) which alternatively updates the rigid transformation and EMD flows in each iteration. To update the rigid transformation, we utilize the *Orthogonal Procrustes* (OP) analysis (Schönemann (1966)).

Suppose the EMD flows $F = \{f_{ij}\}$ are known and we are updating the rigid transformation. Suppose that the two new sets of weighted points are

$$\hat{A} = \left\{a_1^1, \cdots, a_1^{n_2}; a_2^1, \cdots, a_2^{n_2}; \cdots; a_{n_1}^1, \cdots, a_{n_1}^{n_2}\right\}, \tag{3.2}$$

$$\hat{B} = \left\{b_1^1, \cdots, b_{n_2}^1; b_1^2, \cdots, b_{n_2}^2; \cdots; b_1^{n_1}, \cdots, b_{n_2}^{n_1}\right\}, \tag{3.3}$$

where each a_i^j (resp. b_j^i) has the weight f_{ij} and the same spatial position with a_i (resp. b_j). With a slight abuse of notations, a_i^j is also used to denote the corresponding column vector in $\mathbb{R}^d$.

(1) We obtain a translation vector $\mathbf{v}$ satisfying that the weighted centroids of $\hat{A}$ and $\hat{B} + \mathbf{v}$ coincide with each other, which can be easily proved because of the fact that the objective function employs squared distance (Cohen and Guibas (1999)).

(2) Through the OP analysis, we obtain an orthogonal matrix $\mathbf{R}$ for $\hat{B} + \mathbf{v}$ to minimize its weighted ℓ_2^2 distance to $\hat{A}$. For this purpose, we generate two $d \times (n_1 n_2)$ matrices $\mathbf{M}_A$ and $\mathbf{M}_B$, where each point in $\hat{A}$ (resp. $\hat{B} + \mathbf{v}$) corresponds to one column in $\mathbf{M}_A$ (resp. $\mathbf{M}_B$). Specifically, the point $a_i^j \in \hat{A}$ (resp. $b_j^i + \mathbf{v} \in \hat{B} + \mathbf{v}$) corresponds to the column $\sqrt{f_{ij}}\mathbf{a}_i^j$ (resp. $\sqrt{f_{ij}}(\mathbf{b}_j^i + \mathbf{v})$) in $\mathbf{M}_A$ (resp. $\mathbf{M}_B$). Let the singular value decomposition (SVD) of $\mathbf{M}_A\mathbf{M}_B^T$ be $\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$, then by the OP analysis, the optimal orthogonal matrix $\mathbf{R}$ equals $\mathbf{U}\mathbf{V}^T$. In fact, we do not need to explicitly construct the matrices $\mathbf{M}_A$ and $\mathbf{M}_B$, instead, we can compute $\mathbf{M}_A\mathbf{M}_B^T$ in $O\left(n_1 n_2 d + \min\{n_1, n_2\} \cdot d^2\right)$ time (see Lemma 6 for details). Therefore, the time complexity of obtaining the optimal $\mathbf{R}$ is $O\left(n_1 n_2 d + \min\{n_1, n_2\} \cdot d^2 + d^3\right)$.

Lemma 6. *The multiplication* $\mathbf{M}_A\mathbf{M}_B^T$ *can be computed in* $O(n_1n_2d + \min\{n_1, n_2\} \cdot d^2)$ *time.*

Proof. Denote $\mathbf{F}$ as the $n_1 \times n_2$ matrix of the EMD flows where each entry is f_{ij}, and $\mathbf{F}_{i,:}$ represents the i-th row of $\mathbf{F}$. For a vector $\mathbf{f}$, $\sqrt{\mathbf{f}}$ is a new vector whose each entry is the square root of the corresponding one in $\mathbf{f}$. Also, diag($\mathbf{f}$) is a diagonal matrix whose i-th diagonal entry is the i-th entry of $\mathbf{f}$. Following the constructions of $\mathbf{M}_A$ and $\mathbf{M}_B$, we have

$$\begin{aligned}
\mathbf{M}_A &= \left[\sqrt{f_{11}}\mathbf{a}_1^1, \cdots, \sqrt{f_{1n_2}}\mathbf{a}_1^{n_2}, \sqrt{f_{21}}\mathbf{a}_2^1, \cdots, \sqrt{f_{2n_2}}\mathbf{a}_2^{n_2}, \cdots, \sqrt{f_{n_11}}\mathbf{a}_{n_1}^1, \cdots, \sqrt{f_{n_1n_2}}\mathbf{a}_{n_1}^{n_2}\right] \\
&= \left[\mathbf{a}_1\sqrt{\mathbf{F}_{1,:}}, \mathbf{a}_2\sqrt{\mathbf{F}_{2,:}}, \cdots, \mathbf{a}_{n_1}\sqrt{\mathbf{F}_{n_1,:}}\right], \\
\mathbf{M}_B &= \left[\sqrt{f_{11}}\mathbf{b}_1^1, \cdots, \sqrt{f_{1n_2}}\mathbf{b}_{n_2}^1, \sqrt{f_{21}}\mathbf{b}_1^2, \cdots, \sqrt{f_{2n_2}}\mathbf{b}_{n_2}^2, \cdots, \sqrt{f_{n_11}}\mathbf{b}_1^{n_1}, \cdots, \sqrt{f_{n_1n_2}}\mathbf{b}_{n_2}^{n_1}\right] \\
&= \mathbf{B}\left[\mathrm{diag}\left(\sqrt{\mathbf{F}_{1,:}}\right), \mathrm{diag}\left(\sqrt{\mathbf{F}_{2,:}}\right), \cdots, \mathrm{diag}\left(\sqrt{\mathbf{F}_{n_1,:}}\right)\right],
\end{aligned}$$

then

$$\begin{aligned}
\mathbf{M}_A\mathbf{M}_B^T &= \sum_{i=1}^{n_1}(\mathbf{a}_i\sqrt{\mathbf{F}_{i,:}}) \times (\mathrm{diag}(\sqrt{\mathbf{F}_{i,:}})\mathbf{B}^T) \\
&= \sum_{i=1}^{n_1}\mathbf{a}_i\mathbf{F}_{i,:}\mathbf{B}^T \\
&= \mathbf{A}\mathbf{F}\mathbf{B}^T,
\end{aligned}$$

which takes time $O\left(n_1n_2d + \min\{n_1, n_2\} \cdot d^2\right)$. □

Based on the above analyses, we summarize as follows:

Proposition 1. *In (Cohen and Guibas (1999)), each iteration costs*

$$\Gamma(n_1, n_2, d) + O\left(n_1n_2d + \min\{n_1, n_2\} \cdot d^2 + d^3\right),$$

where $\Gamma(n_1, n_2, d)$ *is the time complexity of computing flow matrix. Assume* $n_1, n_2 = O(n)$ *and* $n \geq d$*, then the time complexity is simplified to* $\Gamma(n, d) + O\left(n^2d\right)$.

We notice that the bottleneck in each iteration is large n and d. To reduce the time complexity, we utilize the property of doubling dimension to construct two subsets S_A and S_B of the two

original point sets A and B, and run the same alignment algorithm on S_A and S_B instead. Since n is decreased, the total time complexity is evidently reduced. Moreover, the proposed compression method is independent of the alignment algorithm provided that the alignment algorithm has the same objective function as Definition 6.

3.2 The Proposed Method

In this section, we present the proposed method in detail.

Our method starts with k-center clustering. Given an integer $k \geq 1$ and a point set P in some metric space, k-center clustering is to partition P into k clusters and each cluster is covered by a ball such that the maximum radius of the k balls is minimized. It is known that k-center clustering is NP-hard. Fortunately, (Gonzalez (1985)) proposed an elegant 2-approximation algorithm, which is shown in Algorithm 3, where $\text{dis}(p, S) = \min_{s \in S} \|p - s\|$.

Algorithm 3 2-approximation k-center clustering

Input: Point set P and parameter k
Output: Point set S

1: Choose an arbitrary point c_1 from P and $S \leftarrow \{c_1\}$
2: **for** $i = 2$ to k **do**
3: $\quad c_i \leftarrow \arg\max_{p \in P} \text{dis}(p, S)$
4: $\quad S \leftarrow S \cup \{c_i\}$
5: **end for**

Initially, arbitrarily choose one point c_1 from P and let $S = \{c_1\}$, then iteratively select the point c_i that has the largest distance to S from P and add c_i to S until $|S| = k$.

At first, we denote (X, d_X) as a metric space, where d_X is a metric of set X. A ball centered at point $x \in X$ with radius $r > 0$ is formulated as

$$Ball(x, r) = \{p \in X | d_X(x, p) \leq r\},$$

which is a subset of X.

Claim 1. *Suppose* $S = \{c_1, \cdots, c_k\}$ *and* $r = \min\left\{\left\|c_i - c_j\right\|, 1 \leq i, j \leq k\right\}$*, then* P *can be covered by the* k *balls,* $Ball(c_1, r), Ball(c_2, r), \cdots, Ball(c_k, r)$*.*

Proof. Suppose that there exists such one point $p \in P$ that cannot be covered by any $Ball(c_i, r)$, $1 \leq i \leq k$, then $\|p - c_i\| > r$ and p should be inside S, which is a contradiction. □

Now we introduce the definition of doubling dimension.

Definition 7 (Doubling Dimension)**.** *The doubling dimension of a metric space (X, d_X) is the smallest ρ such that for any $x \in X$ and $r \geq 0$, $Ball(x, r)$ can be always covered by the union of at most 2^ρ balls with radius r.*

For doubling dimension, we have the following claim.

Claim 2. *Let (X, d_X) be a metric space with doubling dimension ρ and $Y \subset X$. If the aspect ratio of Y is upper bounded by α, then $|Y| \leq (2\alpha)^\rho$.*

Here the *aspect ratio* is the ratio of the maximum and the minimum inter-point distances in one point set. Then we have the following lemma.

Lemma 7. *Let P be a point set in $\mathbb{R}^d$ with doubling dimension $\rho \ll d$. The diameter of P is denoted as $\Delta = \max\{\|p - q\| \mid p, q \in P\}$. Given a small parameter $\epsilon > 0$, if one runs Algorithm 3 with $k = \left(\frac{2}{\epsilon}\right)^\rho$, then the radius of each resulting ball is at most $\epsilon\Delta$.*

Proof. Let S be the output of Algorithm 3 and r be the resulting radius, then obviously the aspect ratio of $S \cup \{c_{k+1}\}$ is at most $\frac{\Delta}{r}$. By Claim 2, we have that $k + 1 = \left(\frac{2}{\epsilon}\right)^\rho + 1 \leq \left(\frac{2\Delta}{r}\right)^\rho$ and $r \leq \epsilon\Delta$. □

Let A and B be two given point sets, and the maximum of their diameters be Δ. We also assume that their doubling dimension is ρ. The proposed algorithm for compressing the two original sets A and B is as follows. Set $k = \left(\frac{2}{\epsilon}\right)^\rho$, run Algorithm 3 on A and B and return S_A and S_B respectively. Denote $S_A = \left\{c_1^A, \cdots, c_k^A\right\}$ and $S_B = \left\{c_1^B, \cdots, c_k^B\right\}$. For each cluster center c_i^A (resp. c_i^B), we assign it a weight which equals the total weights of the points belong to this cluster. After that, we obtain a new instance (S_A, S_B) for geometric alignment. Note that the total weights of S_A (resp. S_B) equal W_A (resp. W_B). Our method is shown in Algorithm 4.

Algorithm 4 Geometric Alignment

Input: An instance (A, B) of the geometric alignment problem in Definition 6 with bounded doubling dimension ρ in $\mathbb{R}^d$; parameter $\epsilon \in (0, 1)$.

Output: A rigid transformation $\tilde{\mathcal{T}}$ on B and the EMD flows between A and $\tilde{\mathcal{T}}(B)$.

1: $k \leftarrow \left(\frac{2}{\epsilon}\right)^{\rho}$.
2: Run Algorithm 3 on A and B, and obtain their subsets S_A and S_B respectively.
3: Apply the existing alignment algorithm, e.g., (Cohen and Guibas (1999)) on (S_A, S_B) and obtain the rigid transformation $\tilde{\mathcal{T}}$.
4: Compute the EMD flows between A and $\tilde{\mathcal{T}}(B)$.

The Algorithm 4 is illustrated in Figure 3.5. The proposed algorithm is not only efficient in practice but also has a theoretical upper bound.

Theorem 3. *Suppose $\epsilon > 0$ is a small parameter in Lemma 7. Given constant $c \geq 1$, let $\tilde{\mathcal{T}}$ be a rigid transformation yielding c-approximation of minimizing $\mathcal{EMD}(S_A, \mathcal{T}(S_B))$ in Definition 6. Then we have*

$$\mathcal{EMD}(A, \tilde{\mathcal{T}}(B)) \leq O(c) \cdot \min_{\mathcal{T}} \mathcal{EMD}(A, \mathcal{T}(B)) + O(c\epsilon)\Delta^2. \tag{3.4}$$

Proof. Firstly, we denote

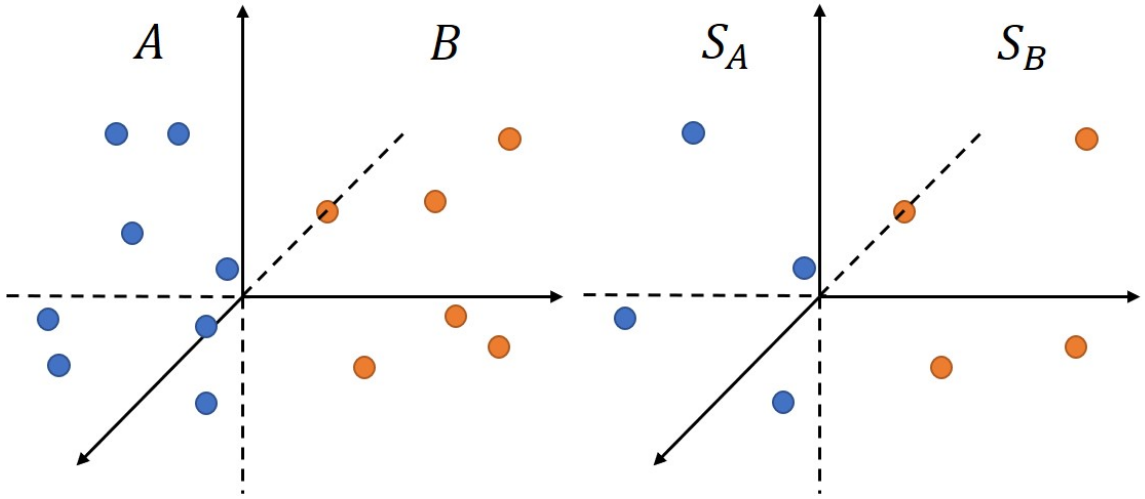
$$\mathcal{T}_{\text{opt}} = \arg\min_{\mathcal{T}} \mathcal{EMD}(A, \mathcal{T}(B)). \tag{3.5}$$

Since $\tilde{\mathcal{T}}$ yields c-approximation of minimizing $\mathcal{EMD}(S_A, \mathcal{T}(S_B))$, we have

$$\begin{aligned} \mathcal{EMD}(S_A, \tilde{\mathcal{T}}(S_B)) &\leq c \cdot \min_{\mathcal{T}} \mathcal{EMD}(S_A, \mathcal{T}(S_B)) \\ &\leq c \cdot \mathcal{EMD}(S_A, \mathcal{T}_{\text{opt}}(S_B)). \end{aligned} \tag{3.6}$$

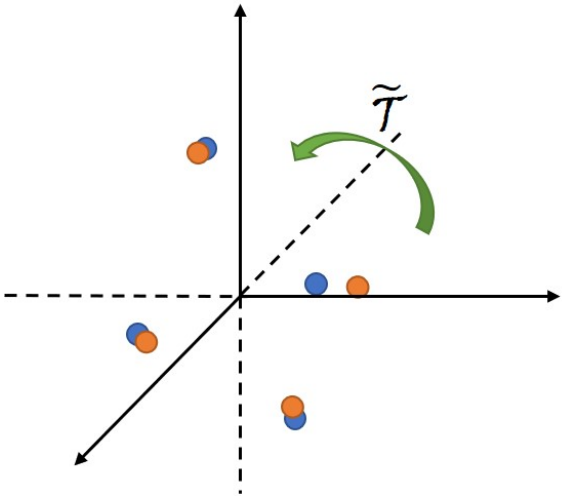
Recall that the weight of each point c_i^A (resp. c_i^B) equals the total weights of the points belonging to c_i^A (resp. c_i^B). For instance, if the corresponding cluster of c_i^A is $\{a_{i_1}, \cdots, a_{i_m}\}$, the weight of c_i^A equals $\sum_{l=1}^{m} \alpha_{i_l}$. Actually, we can view c_i^A as m overlapping points $\left\{a'_{i_1}, \cdots, a'_{i_m}\right\}$ with each $a'_{i_l}, 1 \leq l \leq m$ having the weight α_{i_l}. Therefore, for the sake of convenience, we reformulate S_A and S_B as follows:

$$\begin{aligned} S_A &= \left\{a'_1, \cdots, a'_{n_1}\right\}, \\ S_B &= \left\{b'_1, \cdots, b'_{n_2}\right\}, \end{aligned} \tag{3.7}$$

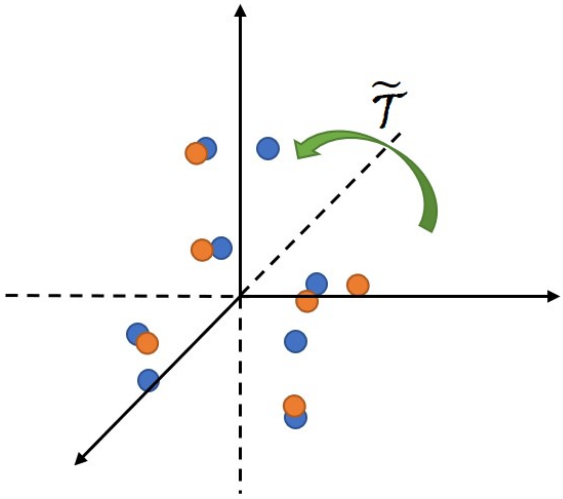


(a) Point sets A and B

(b) Subsets S_A and S_B returned by Algorithm 3



(c) Rigid transformation $\tilde{\mathcal{T}}$ on (S_A, S_B)



(d) Apply $\tilde{\mathcal{T}}$ on (A, B)

Figure 3.5: Illustration of Algorithm 4.

where each a'_i (resp. b'_j) has the weight α_i (resp. β_j). By Lemma 7, we have that for $1 \le i \le n_1, 1 \le j \le n_2$,

$$\begin{aligned}\left\|a'_i - a_i\right\| &\le \epsilon\Delta,\\ \left\|b'_j - b_j\right\| &\le \epsilon\Delta,\end{aligned}$$

and these bounds hold for any rigid transformation in the space. Consequently, for any rigid transformation $\mathcal{T}$ and $1 \le i \le n_1, 1 \le j \le n_2$, by applying triangle inequality we have that

$$\begin{aligned}\left\|a_i - \mathcal{T}(b_j)\right\|^2 &\le \left(\left\|a_i - a'_i\right\| + \left\|a'_i - \mathcal{T}(b'_j)\right\| + \left\|\mathcal{T}(b'_j) - \mathcal{T}(b_j)\right\|\right)^2\\ &\le \left(\left\|a'_i - \mathcal{T}(b'_j)\right\| + 2\epsilon\Delta\right)^2\\ &= \left\|a'_i - \mathcal{T}(b'_j)\right\|^2 + 4\epsilon\Delta\left\|a'_i - \mathcal{T}(b'_j)\right\| + 4\epsilon^2\Delta^2\\ &\le \left\|a'_i - \mathcal{T}(b'_j)\right\|^2 + 2\epsilon\left(\Delta^2 + \left\|a'_i - \mathcal{T}(b'_j)\right\|^2\right) + 4\epsilon^2\Delta^2\\ &= (1+2\epsilon)\left\|a'_i - \mathcal{T}(b'_j)\right\|^2 + (2\epsilon + 4\epsilon^2)\Delta^2.\end{aligned} \tag{3.8}$$

Using the same idea, we have that

$$\begin{aligned}\left\|a'_i - \mathcal{T}(b'_j)\right\|^2 &\le \left(\left\|a'_i - a_i\right\| + \left\|a_i - \mathcal{T}(b_j)\right\| + \left\|\mathcal{T}(b_j) - \mathcal{T}(b'_j)\right\|\right)^2\\ &\le (1+2\epsilon)\left\|a_i - \mathcal{T}(b_j)\right\|^2 + (2\epsilon + 4\epsilon^2)\Delta^2.\end{aligned} \tag{3.9}$$

Based on Definition 4, we denote $\tilde{F} = \{\tilde{f}_{ij}\}$ as the induced flows of $\mathcal{EMD}(S_A, \tilde{\mathcal{T}}(S_B))$ (using expression (3.7) for S_A and S_B). Then (3.8) directly implies that

$$\begin{aligned}\mathcal{EMD}(A, \tilde{\mathcal{T}}(B)) &\le \frac{1}{\min\{W_A, W_B\}}\sum_{i=1}^{n_1}\sum_{j=1}^{n_2}\tilde{f}_{ij}\left\|a_i - \tilde{\mathcal{T}}(b_j)\right\|^2\\ &\le \frac{1+2\epsilon}{\min\{W_A, W_B\}}\sum_{i=1}^{n_1}\sum_{j=1}^{n_2}\tilde{f}_{ij}\left\|a'_i - \tilde{\mathcal{T}}(b'_j)\right\|^2 + (2\epsilon + 4\epsilon^2)\Delta^2\\ &= (1+2\epsilon)\mathcal{EMD}(S_A, \tilde{\mathcal{T}}(S_B)) + (2\epsilon + 4\epsilon^2)\Delta^2.\end{aligned} \tag{3.10}$$

Similarly, let $F = \{f_{ij}\}$ be the induced flows of $\mathcal{EMD}(A, \mathcal{T}_{\text{opt}}(B))$, then (3.9) directly implies that

$$\begin{aligned}
\mathcal{EMD}(S_A, \mathcal{T}_{\text{opt}}(S_B)) &\leq \frac{1}{\min\{W_A, W_B\}} \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} f_{ij} \left\| a'_i - \mathcal{T}_{\text{opt}}(b'_j) \right\|^2 \\
&\leq \frac{1+2\epsilon}{\min\{W_A, W_B\}} \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} f_{ij} \left\| a_i - \mathcal{T}_{\text{opt}}(b_j) \right\|^2 + (2\epsilon + 4\epsilon^2)\Delta^2 \\
&= (1+2\epsilon)\mathcal{EMD}(A, \mathcal{T}_{\text{opt}}(B)) + (2\epsilon + 4\epsilon^2)\Delta^2.
\end{aligned} \tag{3.11}$$

Combining (3.6), (3.10) and (3.11), we can obtain that

$$\begin{aligned}
\mathcal{EMD}(A, \tilde{\mathcal{T}}(B)) &\leq (1+2\epsilon) \cdot \mathcal{EMD}(S_A, \tilde{\mathcal{T}}(S_B)) + (2\epsilon + 4\epsilon^2)\Delta^2 \\
&\leq (1+2\epsilon) \cdot c \cdot \mathcal{EMD}(S_A, \mathcal{T}_{\text{opt}}(S_B)) + (2\epsilon + 4\epsilon^2)\Delta^2 \\
&\leq c(1+2\epsilon)^2 \cdot \mathcal{EMD}(A, \mathcal{T}_{\text{opt}}(B)) + (8c\epsilon^3 + 8c\epsilon^2 + 4\epsilon^2 + 2\epsilon + 2c\epsilon)\Delta^2 \\
&= O(c) \cdot \mathcal{EMD}(A, \mathcal{T}_{\text{opt}}(B)) + O(c\epsilon)\Delta^2,
\end{aligned} \tag{3.12}$$

which completes the proof. □

3.3 Time Complexity Analysis

In this section we analyze the time complexity of Algorithm 4 and consider the steps 2 – 4 individually. To simplify the analysis, we let $n = \max\{n_1, n_2\}$. In step 3, we assume that the alignment algorithm (Cohen and Guibas (1999)) takes λ iterations.

(1) **Step 2.** A straightforward implementation of Algorithm 3 is selecting the k cluster centers iteratively where the resulting running time is $O(knd)$. Several faster implementations for the high dimensional case with low doubling dimension have been studied before; their idea is to maintain some data structures to reduce the amortized complexity of each iteration. We refer readers to (Har-Peled and Mendel (2006)) for more details.

(2) **Step 3.** We run the alignment algorithm (Cohen and Guibas (1999)) on instance (S_A, S_B) instead of (A, B), by Proposition 1, the time complexity of Step 3 is $O\left(\lambda\left(\Gamma(k,d) + k^2d + kd^2 + d^3\right)\right)$.

(3) **Step 4.** In this step, we compute $\mathcal{T}(B)$ first and then $\mathcal{EMD}(A, \mathcal{T}(B))$, which cost $O(nd^2)$ and $\Gamma(n,d)$ respectively.

Note that the complexity $\Gamma(n, d)$ is usually $O(n^2 d)$, which dominates the complexity of Step 2 and Step 4. Based on the above analyses, we have the following theorem.

Theorem 4. *Suppose $n = \max\{n_1, n_2\} \geq d$ and the alignment algorithm (Cohen and Guibas (1999)) takes λ iterations. The running time of Algorithm 4 is*

$$O\left(\lambda\left(\Gamma(k, d) + k^2 d + k d^2 + d^3\right)\right) + \Gamma(n, d).$$

As a contrast, the time complexity of running the same alignment algorithm on the original instance (A, B) is $O\left(\lambda\left(\Gamma(n, d) + n^2 d\right)\right)$ by Proposition 1. When $k \ll n$, Algorithm 4 achieves a significant reduction on the running time.

3.4 Experiments

In this section we implement the proposed method and test the performance on both random and real datasets. All of the experiments are performed on Windows workstation with 2.4GHz Intel Xeon CPU and 32GB DDR4 Memory. For each dataset, we run 20 trials and report the average results. We set the iterative approach (Cohen and Guibas (1999)) to terminate when the change of the objective value is less than 10^{-3}.

3.4.1 Random dataset

To construct a random dataset, we randomly generate two manifolds in $\mathbb{R}^{500}$, which are represented by the polynomial functions with low dimension (≤ 50); in the manifolds, we take two sets of randomly sampled points having the sizes of $n_1 = 2 \times 10^4$ and $n_2 = 3 \times 10^4$ respectively. Also, for each sampled point, we randomly assign a positive weight and finally obtain two weighted point sets as an instance of geometric alignment.

For each instance, we try different compression levels. In Algorithm 4, we set the size of each point set to be k. In experiments, we set $k = \gamma \times \max\{n_1, n_2\}$ where $\gamma \in \{\frac{1}{50}, \frac{1}{40}, \frac{1}{30}, \frac{1}{20}, \frac{1}{10}, 1\}$; in particular, $\gamma = 1$ indicates that we directly run the alignment algorithm (Cohen and Guibas (1999))

without compression. The purpose of our proposed approach is to design a compression method, such that the resulting quantities on the original and compressed datasets are close.

The results of random dataset are shown in Table 3.1. The obtained EMDs on the compressed instances are only slightly higher than the one of $\gamma = 1$, while their running time is significantly reduced. For example, the running time of $\gamma = 1/50$ is less than 5% of the one of $\gamma = 1$.

Table 3.1: Random dataset: EMD and running time of different compression levels.

γ	**1/50**	1/40	1/30	1/20	1/10	**1**
EMD	0.948	0.946	0.945	0.943	0.941	0.933
Time (s)	48.7	54.2	61.0	80.6	144.6	1418.2

To further show the robustness of our method, we particularly add Gaussian noise to the random dataset and study the change of the objective value by varying the noise level. The standard variance of the Gaussian noise is set to be $\eta \times \Delta$, where Δ is the maximum diameter of the point sets and η varies from 0.5×10^{-2} to 2.5×10^{-2}. Figure 3.6 shows that the obtained EMDs over baseline remain very stable (slighly higher than 1).

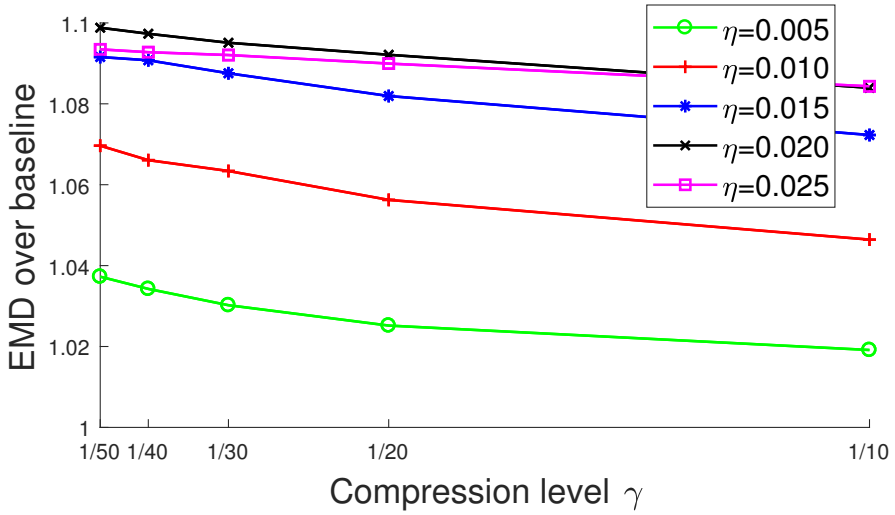


Figure 3.6: The EMDs over baseline for different noise levels.

3.4.2 Real dataset

For real datasets, we consider the two applications mentioned in Section 3.1, unsupervised bilingual lexicon induction and PPI network alignment.

(1) In the first application, we have 5 pairs of languages: *Chinese-English*, *Spanish-English*, *Italian-English*, *Japanese-Chinese*, and *Turkish-English*. Provided by (Zhang et al. (2017)), each language has a vocabulary list that consists of 3000 to 13000 words. We follow their preprocessing idea to represent all the words using vectors in $\mathbb{R}^{50}$ through the embedding technique (Mikolov, Le, and Sutskever (2013)). Actually, each vocabulary list is represented by a distribution in the space where each vector has the weight that equals the corresponding frequency in the language.

(2) In the second application, we use the popular benchmark dataset NAPAbench (Sahraeian and Yoon (2012)) of PPI networks, which contains 3 pairs of PPI networks and each network is a graph of $3000 - 10000$ nodes. As the preprocessing step, we apply the *node2vec* technique (Grover and Leskovec (2016)) to represent each network by a group of vectors in $\mathbb{R}^{100}$, and assign a unit weight to each vector (Liu et al. (2017)).

We obtain the similar performances with the random dataset on the two real datasets whose results are shown in Table 3.2 and Table 3.3 respectively. The EMD for each compression level is always at most 1.2 times the baseline with $\gamma = 1$ while the corresponding runing time is dramatically reduced.

3.5 Summary

In this chapter, we propose a novel framework for compressing point sets in high dimension and simultaneously approximately preserving the quality of geometric alignment. This work is motivated by several emerging applications in the fields of machine learning, bioinformatics and wireless networks. We propose an efficient compression algorithm by virtue of the property of low doubling dimension. What's more, our method has a theoretical upper bound and achieves a much

Table 3.2: Linguistic datasets: EMD and running time of different compression levels.

Datasets \ γ		**1/50**	1/40	1/30	1/20	1/10	**1**
es-en	EMD	0.989	0.969	0.955	0.931	0.892	0.820
	Time (s)	3.4	3.6	3.7	3.9	5.3	100.5
it-en	EMD	0.983	0.966	0.951	0.935	0.899	0.847
	Time (s)	5.4	5.9	6.1	6.5	8.8	162.6
ja-zh	EMD	0.991	0.975	0.962	0.941	0.910	0.836
	Time (s)	1.6	2.1	2.2	2.0	3.0	67.0
tr-en	EMD	0.982	0.960	0.946	0.922	0.889	0.839
	Time (s)	10.1	10.4	11.2	11.9	15.9	287.0
zh-en	EMD	1.014	1.012	0.990	0.962	0.923	0.842
	Time (s)	1.9	1.7	2.2	2.4	2.7	51.6

Table 3.3: PPI network datasets: EMD and running time of different compression levels.

Datasets \ γ		**1/50**	1/40	1/30	1/20	1/10	**1**
CG	EMD	0.0645	0.0644	0.0643	0.0643	0.0642	0.0642
	Time (s)	2.3	1.8	2.3	1.9	2.5	7.6
DMC	EMD	0.0642	0.0641	0.0641	0.0639	0.0639	0.0638
	Time (s)	3.0	2.9	3.0	3.0	3.1	11.4
DMR	EMD	0.0567	0.0566	0.0564	0.0564	0.0563	0.0562
	Time (s)	2.3	2.1	2.4	2.1	2.7	13.7
dm-mm	EMD	0.1344	0.1343	0.1343	0.1342	0.1342	0.1341
	Time (s)	0.6	0.6	0.6	0.7	0.8	2.0
hs-mm	EMD	0.0774	0.0773	0.0773	0.0773	0.0772	0.0772
	Time (s)	1.4	1.5	1.5	1.6	1.8	4.3

smaller time complexity. Experimental results indicate that our method can significantly speed up the existing alignment algorithms while preserve the alignment quality.

BIBLIOGRAPHY

BIBLIOGRAPHY

Achlioptas, D. 2003. Database-friendly random projections: Johnson-lindenstrauss with binary coins. *Journal of computer and System Sciences* 66(4):671–687.

Agarwal, P. K.; Fox, K.; Panigrahi, D.; Varadarajan, K. R.; and Xiao, A. 2019. Faster algorithms for the geometric transportation problem. *arXiv preprint arXiv:1903.08263*.

Agarwal, P. K.; Har-Peled, S.; and Varadarajan, K. R. 2005. Geometric approximation via coresets. *Combinatorial and Computational Geometry* 52:1–30.

Agarwal, P. K.; Har-Peled, S.; and Yu, H. 2008. Robust shape fitting via peeling and grating coresets. *Discrete & Computational Geometry* 39(1-3):38–58.

Aggarwal, C. C., and Yu, P. S. 2001. Outlier detection for high dimensional data. *ACM Sigmod Record* 30(2):37–46.

Andoni, A.; Do Ba, K.; Indyk, P.; and Woodruff, D. 2009. Efficient sketches for earth-mover distance, with applications. In *2009 50th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 324–330.

Badoiu, M., and Clarkson, K. L. 2003. Smaller core-sets for balls. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 801–802.

Bădoiu, M., and Clarkson, K. L. 2008. Optimal core-sets for balls. *Computational Geometry* 40(1):14–22.

Badoiu, M.; Har-Peled, S.; and Indyk, P. 2002. Approximate clustering via core-sets. In *Proceedings of the ACM Symposium on Theory of Computing (STOC)*, 250–257.

Belkin, M. 2003. Problems of learning on manifolds.

Besl, P. J., and McKay, N. D. 1992. Method for registration of 3-d shapes. In *Sensor Fusion IV: Control Paradigms and Data Structures*, volume 1611, 586–607.

Blum, M.; Floyd, R. W.; Pratt, V.; Rivest, R. L.; and Tarjan, R. E. 1973. Time bounds for selection. *Journal of Computer and System Sciences* 7(4):448–461.

Breunig, M. M.; Kriegel, H.-P.; Ng, R. T.; and Sander, J. 2000. Lof: identifying density-based local outliers. *ACM Sigmod Record* 29(2):93–104.

Cabello, S.; Giannopoulos, P.; Knauer, C.; and Rote, G. 2008. Matching point sets with respect to the earth mover's distance. *Computational Geometry* 39(2):118–133.

Cao, X.; Wei, Y.; Wen, F.; and Sun, J. 2014. Face alignment by explicit shape regression. *International Journal of Computer Vision* 107(2):177–190.

Chandola, V.; Banerjee, A.; and Kumar, V. 2009. Anomaly detection: A survey. *ACM Computing Surveys (CSUR)* 41(3):15.

Clarkson, K. L. 2010. Coresets, sparse greedy approximation, and the frank-wolfe algorithm. *ACM Transactions on Algorithms* 6(4):63.

Cohen, S., and Guibas, L. 1999. The earth mover's distance under transformation sets. In *Proceedings of the Seventh IEEE International Conference on Computer Vision (ICCV)*, volume 2, 1076–1083.

Dasgupta, S., and Sinha, K. 2013. Randomized partition trees for exact nearest neighbor search. In *Conference on Learning Theory (COLT)*, 317–337.

Ding, H., and Xu, J. 2011. Solving the chromatic cone clustering problem via minimum spanning sphere. In *Proceedings of the International Colloquium on Automata, Languages, and Programming (ICALP)*, 773–784.

Ding, H., and Xu, J. 2014. Sub-linear time hybrid approximations for least trimmed squares estimator and related problems. In *Proceedings of the International Symposium on Computational geometry (SoCG)*, 110.

Ding, H., and Xu, J. 2015. Random gradient descent tree: A combinatorial approach for svm with outliers. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2561–2567.

Ding, H., and Xu, J. 2017. Fptas for minimizing the earth mover's distance under rigid transformations and related problems. *Algorithmica* 78(3):741–770.

Ester, M.; Kriegel, H.-P.; Sander, J.; and Xu, X. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 226–231.

Fei-Fei, L.; Fergus, R.; and Perona, P. 2007. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. *Computer vision and Image understanding* 106(1):59–70.

Frank, M., and Wolfe, P. 1956. An algorithm for quadratic programming. *Naval Research Logistics Quarterly* 3(1-2):95–110.

Gärtner, B., and Jaggi, M. 2009. Coresets for polytope distance. In *Proceedings of the International Symposium on Computational geometry (SoCG)*, 33–42.

Gilbert, E. G. 1966. An iterative procedure for computing the minimum of a quadratic form on a convex set. *SIAM Journal on Control* 4(1):61–80.

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

Grover, A., and Leskovec, J. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining (KDD)*, 855–864.

Gupta, M.; Gao, J.; Aggarwal, C.; and Han, J. 2014. Outlier detection for temporal data. *Synthesis Lectures on Data Mining and Knowledge Discovery* 5(1):1–129.

Har-Peled, S., and Mendel, M. 2006. Fast construction of nets in low-dimensional metrics and their applications. *SIAM Journal on Computing* 35(5):1148–1184.

Har-Peled, S., and Wang, Y. 2004. Shape fitting with outliers. *SIAM Journal on Computing* 33(2):269–285.

Hinton, G. E.; Srivastava, N.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. R. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

Indyk, P., and Thaper, N. 2003. Fast image retrieval via embeddings.

Indyk, P. 2007. A near linear time constant factor approximation for euclidean bichromatic matching (cost). In *SODA*, volume 7, 39–42.

Karger, D. R., and Ruhl, M. 2002. Finding nearest neighbors in growth-restricted metrics. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing (STOC)*, 741–750.

Klein, O., and Veltkamp, R. C. 2005. Approximation algorithms for computing the earth mover's distance under transformations. In *International Symposium on Algorithms and Computation (ISAAC)*, 1019–1028.

Krauthgamer, R., and Lee, J. R. 2004. Navigating nets: simple algorithms for proximity search. In *Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms (SODA)*, 798–807.

Kriegel, H.-P.; Kröger, P.; Schubert, E.; and Zimek, A. 2009. Outlier detection in axis-parallel subspaces of high dimensional data. In *Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*, 831–838.

Kriegel, H.-P.; Kröger, P.; and Zimek, A. 2009. Outlier detection techniques. *Tutorial at PAKDD*.

Kriegel, H.-p.; Schubert, M.; and Zimek, A. 2008. Angle-based outlier detection in high-dimensional data. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 444–452.

Kumar, P.; Mitchell, J. S. B.; and Yildirim, E. A. 2003. Approximate minimum enclosing balls in high dimensions using core-sets. *ACM Journal of Experimental Algorithmics* 8.

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

LeCun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86(11):2278–2324.

Liu, Y.; Ding, H.; Chen, D.; and Xu, J. 2017. Novel geometric approach for global alignment of ppi networks. In *Thirty-First AAAI Conference on Artificial Intelligence (AAAI)*.

Liu, W.; Hua, G.; and Smith, J. R. 2014. Unsupervised one-class learning for automatic outlier removal. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 3826–3833.

Lu, C.; Shi, J.; and Jia, J. 2013. Abnormal event detection at 150 fps in matlab. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2720–2727.

Magnanti, T., and Orlin, J. 1993. Network flows: theory, algorithms and applications.

Malod-Dognin, N.; Ban, K.; and Pržulj, N. 2017. Unified alignment of protein-protein interaction networks. *Scientific reports* 7(1):953.

Maltoni, D.; Maio, D.; Jain, A. K.; and Prabhakar, S. 2009. *Handbook of fingerprint recognition*. Springer Science & Business Media.

Mikolov, T.; Le, Q. V.; and Sutskever, I. 2013. Exploiting similarities among languages for machine translation. *arXiv preprint arXiv:1309.4168*.

Munteanu, A., and Schwiegelshohn, C. 2018. Coresets-methods and history: A theoreticians design pattern for approximation and streaming algorithms. *KI-Künstliche Intelligenz* 32(1):37–53.

Phillips, J. M. 2016. Coresets and sketches. *Computing Research Repository*.

Rumelhart, D. E.; Hinton, G. E.; and Williams, R. J. 1988. Learning representations by back-propagating errors. *Cognitive Modeling* 5(3):1.

Sahraeian, S. M. E., and Yoon, B.-J. 2012. A network synthesis model for generating protein interaction network families. *PloS one* 7(8):e41474.

Sakurada, M., and Yairi, T. 2014. Anomaly detection using autoencoders with nonlinear dimensionality reduction. In *Proceedings of the Workshop on Machine Learning for Sensory Data Analysis (MLSDA)*, 4.

Schölkopf, B.; Williamson, R.; Smola, A.; Shawe-Taylor, J.; and Platt, J. 1999. Support vector method for novelty detection. In *Proceedings of the Advances in Neural Information Processing Systems (NIPS)*, 582–588.

Schönemann, P. H. 1966. A generalized solution of the orthogonal procrustes problem. *Psychometrika* 31(1):1–10.

Simonyan, K., and Zisserman, A. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Song, L.; Ma, H.; Wu, M.; Zhou, Z.; and Fu, M. 2018. A brief survey of dimension reduction. In *International Conference on Intelligent Science and Big Data Engineering (IScIDE)*, 189–200.

Talwar, K. 2004. Bypassing the embedding: algorithms for low dimensional metrics. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing (STOC)*, 281–290.

Tan, P.-N.; Steinbach, M.; and Kumar, V. 2006. *Introduction to Data Mining*.

Tenenbaum, J. B.; De Silva, V.; and Langford, J. C. 2000. A global geometric framework for nonlinear dimensionality reduction. *science* 290(5500):2319–2323.

Xia, Y.; Cao, X.; Wen, F.; Hua, G.; and Sun, J. 2015. Learning discriminative reconstructions for unsupervised outlier removal. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 1511–1519.

Youn, H.; Sutton, L.; Smith, E.; Moore, C.; Wilkins, J. F.; Maddieson, I.; Croft, W.; and Bhattacharya, T. 2016. On the universal structure of human lexical semantics. *Proceedings of the National Academy of Sciences* 113(7):1766–1771.

Zarrabi-Zadeh, H., and Mukhopadhyay, A. 2009. Streaming 1-center with outliers in high dimensions. In *Proceedings of the Canadian Conference on Computational Geometry (CCCG)*, 83–86.

Zhang, M.; Liu, Y.; Luan, H.; and Sun, M. 2017. Earth mover's distance minimization for unsupervised bilingual lexicon induction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 1934–1945.