Distributed Distance Learning Algorithms and Applications

SU, Yuxin

A Thesis Submitted in Partial Fulfilment
of the Requirements for the Degree of
Doctor of Philosophy
in
Computer Science and Engineering

The Chinese University of Hong Kong
September 2019

Thesis Assessment Committee

Professor CHENG James (Chair)

Professor LYU Rung Tsong Michael (Thesis Supervisor)

Professor KING Kuo Chin Irwin (Thesis Co-supervisor)

Professor YOUNG Fung Yu (Committee Member)

Professor KWONG Sam Tak Wu (External Examiner)

Abstract of thesis entitled:

Distributed Distance Learning Algorithms and Applications

Submitted by SU, Yuxin

for the degree of Doctor of Philosophy

at The Chinese University of Hong Kong in September 2019

Distance learning, also called distance metric learning is an effective similarity learning tool to learn a distance function from examples to enhance the performance of machine learning models in applications of classification, regression, and ranking and so on. Most distance learning algorithms involve a positive semi-definite matrix as critical parameters that scales quadratically with the number of dimensions of input data. This situation brings tremendous computational cost in the learning procedure and makes all proposed algorithms infeasible for extremely high-dimensional data even with the low-rank approximation. In this thesis, we consistently explore the computational complexity of distance learning algorithms from multiple perspectives. Our exploration ranges from the linear to non-linear models, from the shallow to deep models, and from the deterministic to probabilistic models. Specifically, we propose several distributed algorithms to enhance the distance learning algorithms performance in terms of computational speed and learning accuracy.

Firstly, we focus on linear distance learning algorithms. To parallel the popular Information-Theoretic Metric Learning (ITML) with tolerated errors, we utilize the property that each positive semi-definite matrix can be decomposed into a combination of rank-one and trace-one matrices and convert the original sequential training procedure into a parallel one. In most cases, the communication demands of the proposed method are also reduced from $\mathcal{O}(d^2)$ to $\mathcal{O}(cd)$, where d is the number of dimensions of the data and c is the number of constraints in linear distance learning and can be smaller than d by appropriate selection. Moreover importantly, we present a rigorous theoretical analysis to upper bound the Bregman divergence between the sequential algorithm and the parallel algorithm, which guarantees the correctness and performance of the proposed algorithm.

Secondly, we continue to explore non-linear distance learning algorithms. To capture the local structures in non-linear metric space, we propose a novel parallel Riemannian metric learning algorithm. Then we successfully apply the proposed algorithms on the classical learning-to-rank problem to achieve the state-of-the-art in learning-to-rank community.

Thirdly, we explore the distance learning algorithms from the shallow models to deep distance learning models, which is widely used in extreme classification and image retrieval because of its powerful ability to learn the semantic low-dimensional embedding of high-dimensional data. However, the high computational cost of evaluating mini-batch and updating models frequently in existing deep distance learning approaches becomes a barrier to apply such methods to a large-scale real-world context in a distributed environment. In this thesis, we introduce a novel distributed framework to speed up the training process of deep distance learning using multiple machines. Specifically, we design a hybrid communication pattern and implement a decentralized data-parallel framework to reduce the communication time while the quality of the trained deep distance models is preserved.

Finally, we explore the distance learning algorithm in the probabilistic domain. In probability theory, Wasserstein distance is used to estimate the similarity between two distributions. However, the existing formulations with approximated Wasserstein loss converge slowly due to substantial computation cost and usually generate unstable results as well. In this thesis, we attempt to solve the computation cost problem by speeding up the computation of Wasserstein distance from a well-designed communication efficient parallel architecture. Then, we also employ a new formulation and the corresponding optimization method to improve the stability with a more accurate Wasserstein distance estimation. Compared to conventional parallel architecture for deep neural networks, our proposed framework enjoys a higher scalability performance.

論文題目：分布式距離學習算法與應用
作者　　：蘇玉鑫
學校　　：香港中文大學
學系　　：計算機科學與工程學系
修讀學位：哲學博士
摘要　　：

距离學習，又稱距離度量學習，是一種有效的相似性學習工具。它從數據中學習更有效的距離函數，用於提升分類、回歸、排序等機器學習模型的性能。傳統的距離學習方法會涉及計算複雜度極高的半正定矩陣計算，這無疑給機器學習過程帶來巨大的計算負擔，且難以應用於高維數據中。在本論文中，我們從多個層面探索距離學習的計算複雜度問題，包拓線性模型、非線性模型、深度學習模型、概率距離模型。基於複雜度分析，我們提出一系列分布式方法用於提升距離學習模型的計算速度與精度。

第一部分，關於線性距離學習算法，我們利用半正定矩陣可以被分解單秩單跡矩陣的組合這一特性，將流行的信息論度量學習算法由串行算法改造為並行算法，並同時降低此並行算法的通信複雜度。最後我們通過嚴密的理論分析，給出了原始的串行算法與提出的並行算法的誤差上限。

第二部分，我們繼續探索非線性距離學習。通過挖掘非線性距離度量空間中的局部結構，我們提出了一種有效的並行黎曼度量學習方法，並成功應用於排序問題。

第三部分，我們探索深度度量學習模型。由於此模型可以從高維數據中學習具有語義含義的低維表示，因而它泛應用於圖像

分類與圖像檢索任務。然而傳統深度度量學習的高計算複雜度問題使得此類方法無法以分布式的方式處理真實世界中大規模數據問題。在本論文中，我們介紹了一種新穎的分布式計算方法，可以有效解決此難題。與此同時，我們通過設計一種混合的去中心化的通信機制，在保證學習精度的情況下可以顯著降低通信時間。

最後，我們試圖探索將分布式計算方法應用於概率分布距離的計算當中。在概率論中，Wasserstein距離是有效的估計概率分布距離的方法。然而，已有的方法大多基於近似計算，不僅收斂速度較慢，還會產生不穩定的計算結果。在此論文中，我們提出一種新型的Wasserstein距離的計算形式和與此對應的優化方法，並設計了一種高效的分布式計算策略，可以顯著提升計算精度與計算穩定性。

Acknowledgement

First and foremost, I would like to thank my supervisors, Prof. Michael R. Lyu and Prof. Irwin King for their excellent supervision during my Ph.D. study at CUHK. From choosing the research topic to technical writing, their inspiring guidance and patience help me conduct tough research work. During the long Ph.D. study period, I have learned so much from their knowledge and attitude in doing research.

I am grateful to my thesis assessment committee members, Prof. James Cheng, Prof. Evangeline Young for their constructive comments and valuable suggestions to this thesis and all my term presentations. Great thanks to Prof. Tak Wu Kwong from City University of Hong Kong who kindly serves as the external examiner for this thesis.

I would like to thank my mentor, Mr. Taifeng Wang when I intern in Microsoft Research Asia for the insightful discussions on distributed machine learning algorithms. Also, I would like to thank Dr. Shenglin Zhao and Dr. Xixian Chen when I intern in Tencent Youtu Lab for their valuable contributions to the research in this thesis.

I would like to thank Prof. Haiqin Yang and Prof. Yangfan Zhou for their insightful discussion and inspiring guidance on the research topic in the early stage of my Ph.D. study.

I am also thankful to my fantastic group fellows, Yu Kang, Guang Ling, Chen Cheng, Jieming Zhu, Hongyi Zhang, Junjie Hu, Pinjia He, Tong Zhao, Xiaotian Yu, Hui Xu, Cuiyun Gao, Jichuan Zeng, Jiani Zhang, Ken Chan, Jian Li, Han Shao, Wang Chen, Shilin He, Yue Wang, Pengpeng Liu, Haoli Bai, Wenxiang Jiao, Yifan Gao, Jingjing Li, Weibin Wu, Zhuangbin Chen and Tianyi Yang.

Last but most important, I would like to thank my wife Ms. Yingchen Cai and my parents. Their deep love and constant support are the driving force during my Ph.D. study.

To my family.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview

Similarity and the associated distance measurements are the fundamental components in machine learning and data mining techniques. For instance, in classification, the k-Nearest Neighbor classifier employs a distance measurement to find similar pairs and dissimilar pairs of samples to identify the nearest neighbors. Many clustering algorithms, such as the prominent k-means, rely on distance measurements between data points.
This kind of algorithms has usually used a general-purpose distance to measure the data similarity. The performance of these methods depends on the quality of the similarity metrics. We hope that the similar pairs identified by the distance measurements should be semantically close. Meanwhile, the dissimilar pairs should be semantically different. Therefore, the construction of distance learning algorithms should rely on categorical information. Currently, the widely used distances include the Euclidean distance, the cosine similarity for feature vectors, and the Levenshtein distance for strings. However, a standard distance may ignore some task-specific properties

available in the training dataset and often fail to capture the unique characteristics of the data, so that the learning results could be sub-optimal or even lead to the wrong direction.

Therefore, a fundamental question that we need to consider is how to assess the similarity or distance between the pairs of data samples precisely. In data mining problems, the definition of similarity is often task-specific. The success of distance learning hinges on aligning its learning objectives with the task. Therefore, when the distance function is designed for a specifical task, the results of data mining algorithms should be improved. Unfortunately, for specific problems, it is difficult to find a unique and appropriate way to measure the distance or similarity between data, which is ubiquitous in machine learning, pattern recognition, and data mining. To handle multiple similarity or distance functions, we should determine by hand an appropriate choice of features and the combination of those features. This motivates the emergence of distance learning, which aims at automatically learning a distance from data and has attracted considerable interest in machine learning and data mining community for the past two decades.

Distance learning algorithms attempt to automate the process of feature selection and learn task-specific distance functions based on the data in a supervised manner. These algorithms have been shown to be useful when used in conjunction with nearest-neighbor methods and other techniques like regression, and ranking that rely on distances or similarities.

There is another popular technique called dimensionality reduction, which is similar to distance learning. However, they differ in the following perspectives: (a), for supervised dimensionality reduction, it aims at finding a low dimensional representation that maximizes the separation of labeled data and in this respect has connections with distance learning, although the primary objective is quite different. (b), for unsupervised dimensionality reduction, it usually assumes that the input data lie on the surface of low-dimensional manifold within the higher-dimensional space. These methods are designed to capture or preserve some properties of the original data (such as the variance or local distance measurements) in the low-dimensional representation. For high-dimensional data, distance learning is sometimes formulated as finding a projection of the data into a new feature space, which preserves the useful geometrical information of the samples.

In distance learning framework, we assume the accurate target distance between pairs of instances is intangible. The general supervision comes from two perspectives: a), pair-wise constraints: the pair of data x_i and x_j is similar, the pair of data x_p and x_q is dissimilar. b), triplet-wise constrants: the data x_i is more similar to x_j than to x_k. Compared to conventional pair-wise distance learning framework, triplet-wise distance learning allows circumventing the use of highly sensitive parameters such as distance thresholds and margins.

The distance functions can be parameterized in various ways. Global distance learning methods aim to learn a single mapping f to be applied to all the data. The widely used principal components analysis (PCA) method is considered as seeking a linear map in an unsupervised manner to be applied globally to all the data. Conventional distance learning methods usually employ the Mahalanobis distance or kernel-based distance function to formulate this linear or non-linear mapping.

Typically, most popular distance learning methods usually learn a linear mapping to project samples into a new feature space. When the non-linear relationship of data points is considered, the kernel method is adopted to address this non-linearity problem. While this type of method suffers from the scalability problem because the kernel-based method has two major disadvantages: a), choosing a kernel is typically tricky and entirely empirical. b), the expression power of kernel functions is often not flexible enough to capture the non-linearity in the data.

A single global distance function is often unable to describe the complex non-linear space in the specific task. Investigations on local distance learning have considered locality specific approaches, and consequently multiple distance functions are learned. Specially, multiple distance function learning techniques learn local metric tensors in different parts of feature space. Armed with multiple distance functions, even simple classifiers can be competitive with the state-of-the-art because each distance function locally formulates the structure of the data. However, the learned combination of multiple distance functions is, non-metric in mathematical, which has prevented multi-distance learning from generalizing to tasks such as dimensionality reduction and regression in an elegant way.

Moreover, from the perspective of manifold learning, all local distance learning can be regarded as approximations of the geodesic distance defined by a metric tensor at the appropriate points. This idea ushers us to rethink the distance learning algorithms with the help of deep learning framework in a more elegant way.

Mathematically, a manifold can be effectively used to model the non-linearity of samples in the training set, and deep learning has demonstrated the superb capability to model the non-linearity of samples. Deep neural network can be considered to directly learn the hierarchical non-linear mapping function from the input data to a lower- dimensional embedding given the input label annotations. The advantage is that the training process can jointly learn the distance learning function and semantically meaningful embedding, which are robust against intra-class variations [106].

In the semantic embedding framework, similar examples are mapped close to each other, and dissimilar examples are mapped farther apart. A contrastive loss over the distance function induced by the semantic representation is employed to train the network to distinguish between similar and dissimilar pairs of examples. Theoretically, the embedded space should be described by a positive semi-definite matrix. However, the back propagation algorithm in traditional deep learning can not preserve such form of constraints [50].

Different from traditional distance learning framework, we employ deep neural networks with distance learning based loss function to learn discriminative embedding instead of directly modeling distance function. The contrastive loss is designed to bring the embedding of samples from the same category closers, to separate the embedding of samples from different classes with a reasonable margin. When the embedding space is appropriately obtained, the Euclidean distance in this embedding space should reflect the actual semantic distance between data points.

Previous distance learning algorithms focus on the similarity measurement of the deterministic feature of data samples. However, the world is full of randomness. We should solve the complex challenge brought by any random issue and develop reliable similarity measurement between probability distributions. In mathematical theory, distance measurements between probability distributions can be introduced in the framework of optimal transport. For deterministic distance learning framework, algorithms will perform a point-wise comparison between two distributions. In optimal transport, the distance between two distributions is the minimal effort for moving the probability mass of one distribution to the other. While the transport plan to move the mass is optimized according to a given ground metric space. When the ground metric space is p-norm, this distance is also called as Wasserstein-p distance. Typically, the Wasserstein distance is regarded as a loss function for unsupervised learning. The algorithmic performance mainly depends on the choice of the ground metric on the input data space.

Distance learning is applicable to a wide range of different machine learning tasks such as information retrieval [77, 91], computer vision [110], recommendation system [17, 83], sentiment analysis [130]. For example, nearest neighbor retrieval is the most important application area of distance learning. The objective of many information retrieval systems, such as search engines, is to provide the user with the most relevant documents according to his/her query. Specifically, documents are ranked according to their relevance to a given query based on similarity scores [55, 76, 84]. This ranking is often achieved by using a distance learning between two documents or between a document and a query.

In computer vision, there is a great need of well-designed distance functions, not only to compare images or videos in semantic representations such as bags-of-visual-words but also in the pre-processing step targeting to build this semantic representation. For this reason, there exists a large body of distance learning literature designed specifically to solve computer vision problems (e.g. image classification [85], object recognition [115], vision tracking [60]).

Since the amount of available data growing fast, the problem of scalability arises in all areas of machine learning. Among different kinds of machine learning algorithms, the barrier of scalability in distance learning is urgently necessary to overcome. First, in terms of big data, it is desirable for a distance learning algorithm to scale well with the number of training examples n or semi-supervised constraints. Under this situation, learning the distance function in an online way is one of the solutions. Second, and more importantly, distance learning methods should also scale reasonably well with the dimensionality d of the data, which is also classified into the big model problem. However, since conventional distance learning is often formulated as learning a $d \times d$ matrix, designing algorithms that scale reasonably well with this quantity is a huge challenge. For specific algorithms, The first one is to main $M \in \mathbb{S}_+^d$ in an efficient way during the optimization process. In optimization thoery, a straightforward way to conduct this task is to employ the projected gradient method which consists in alternating between a gradient step and a projection step onto the PSD cone by setting the negative eigenvalues to zero [92]. However, the projection operation is prohibitively expensive for high-dimensional problems because eigenvalue decomposition scales in $\mathcal{O}(d^3)$. Another challenge is to learn a low-rank matrix which implies a low-dimensional projection space instead of a costly full-rank matrix. Unfortunately, the optimization subject to a

rank constraint or regularization about M is NP-hard and thus cannot be computed out efficiently.
Compared with the classification task, deep distance learning methods process much more training samples since all possible combinations of data need to be considered. Although we can construct a large number of data pairs for deep distance learning, a significant fraction of non-informative pairs will contribute zero to the loss function and gradient once the model reaches a reasonable performance. In this situation, deep distance learning is known to suffer from slow convergence. Thus it is intuitive to find more informative pairs during training to achieve faster convergence and better performance. For large scale probability problems, the accurate computation of Wasserstein distance involves solving a linear program whose cost quickly becomes prohibitive while the data dimension increases.

1.2 Thesis Contributions

In this thesis, we mainly focus on employing parallel or distributed computation techniques to solve the above big data challenges in the distance learning framework. Specifically, we resolve this problem and make contributions from multiple perspectives including global distance learning with linear transformation, non-linear distance learning with local combinations, highly non-linear distance learning with deep neural networks and probabilistic distance learning with uncertainty.

- For global distance learning framework, we first propose a parallel approach to learn distance functions from high-dimensional data without restricting low-rank assumption and achieve a balance between accuracy and execution time. Then, we compare the proposed method with the popular ITML method from a theoretical perspective and give an upper error bound brought by our parallel update. Finally, we conduct rigorous experiments on the data with different scales demonstrate the correctness and the scalability of our proposed method.

- For local distance learning, we are the first to extend the geometric mean metric learning algorithm to a local distance learning approach in order to capture the local structures for the learning-to-rank problem. We propose a novel *ideal candidate document* concept to transform distance-learning-to-rank framework from query dependent model to query independent model, which brings broader applications for distance learning and also improves the accuracy of classical learning-to-rank task. Finally, we conduct extensive experiments to demonstrate that our method outperforms the state-of-the-art query-dependent distance-learning-to-rank algorithms and query-independent learning-to-rank methods both in the accuracy and the computational complexity.

- For deep distance learning framework, we are the first to propose a distance learning-oriented distributed framework to speed up the training of deep distance learning among multiple machines. We have verified the decentralized distributed computation with mixed communication topology is reasonable in the context of the deep distance learning framework. Finally, we also demonstrate the proposed framework is appropriately coupled with several state-of-the-art deep distance learning algorithms on CUB200-2011, CARS196, and Stanford Online Products and achieves a remarkable improvement with four machines regarding accuracy and runtime speedup.

- For the distance learning with probability distributions, we are the first to propose a parallel architecture for Wasserstein GANs framework to speed up the training of GANs from the computational perspective. Different from common Wasserstein GANs, we develop an efficient stochastic algorithm to approximate the Wasserstein distance with higher accuracy for the newly proposed parallel framework. In experiments, we show that the proposed parallel architecture enjoys the superior convergence speed and comparable stability.

1.3 Thesis Organization

The remainder of this thesis is organized as follows.

- **Chapter 2**
  In this chapter, we provide a systematic review of the background of distance learning algorithms, including global distance learning, local distance learning, deep distance learning, and learning distances between probability distributions.

- **Chapter 3**
  In this chapter, we introduce a semi-synchronous distributed methodology for global distance learning algorithms. More specifically, in section 3.3, we present a rigorous theoretical analysis to upper bound the Bregman divergence between the sequential algorithm and the parallel

algorithm. In section 3.4, we demonstrate the competitive scalability and the performance compared with the original ITML algorithm on datasets with $\mathcal{O}(10^5)$ features.

- **Chapter 4**
  In this chapter, we propose a novel Riemannian metric learning algorithm to capture the local structures and develop a robust learning-to-rank algorithm. In section 4.3, we demonstrate that our proposed L-GMML algorithm outperforms the state-of-the-art distance learning to rank methods and the stylish query-independent learning-to-rank algorithms regarding the accuracy and computational efficiency.

- **Chapter 5**
  In this chapter, we introduce a novel distributed framework to speed up the training process of the deep distance learning using multiple machines. In section 5.3, we show excellent performance gain compared to a full spectrum of state-of-the-art deep distance learning models on multiple datasets in terms of image clustering and image retrieval tasks.

- **Chapter 6**
  In this chapter, we introduce a well-designed communication efficient parallel architecture to solve the computation cost problem by speeding up the Wasserstein GANs. In section 6.3, we demonstrate rigorous experiments to reveal that our proposed framework achieves a significant improvement regarding convergence speed with comparable stability on generating images, compared to the state-of-the-art of Wasserstein GANs algorithms.

- **Chapter 7**
  The last chapter summarizes this thesis and discusses some potential future research directions about distance learning algorithms.

□ **End of chapter.**

Chapter 2

Background Review

2.1 Distance Learning Algorithms

In mathematics, distance function denoted as $f : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ should satisfy the following basic rules:

- **non-negativity**: $f(x,y) \geq 0$
- **identity**: $f(x,y) = 0 \Longleftrightarrow x = y$
- **symmetry**: $f(x,y) = f(y,x)$
- **triangle inequality**: $f(x,z) \leq f(x,y) + f(y,z)$

A plenty of non-parameter distance functions satisfy the above rules, e.g. Euclidean distance $f(x,y) = \sqrt{(x-y)^T(x-y)}$, Cosine distance $f(x,y) = 1 - \frac{x^T y}{|x||y|}$. Another popular distance functions may not satisfy all these rules, e.g. Kullback-Leibler (KL) divergence $f(x,y) = \sum_i x_i \log \frac{x_i}{y_i}$.
The above distance functions are general for all machine learning tasks. Difference with these non-parametric distance function, the goal of distance learning is to adapt some pairwise real-valued distance function, like the Mahalanobis distance in

Eq. (2.1), to the problem of interest using the information brought by training examples.

$$f_M(\mathbf{x},\mathbf{x}') = \sqrt{(\mathbf{x}-\mathbf{x}')^T M(\mathbf{x}-\mathbf{x}')} \tag{2.1}$$

Specifically, given n instances $\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\}$ in $\mathbb{R}^d$, we aims at learning the matrix M which is involved in the Mahalanobis distance function.

In the literature, most classical methods learn the distance function with the positive semi-definite matrix M in f_M in a weakly-supervised way from pair-wise or triplet-wise constraints of the following forms:

- Positive / negative pairs (Must-link / cannot-link constraints)

$$\mathcal{S} = \{(x_i, x_j) \ : \ x_i \text{ and } x_j \text{ should be similar}\} \tag{2.2}$$
$$\mathcal{D} = \{(x_i, x_j) \ : \ x_i \text{ and } x_j \text{ should be dissimilar}\} \tag{2.3}$$

- Relative constraints (training triplets)

$$\mathcal{R} = \{(x_i, x_j, x_k) \ : \ x_i \text{ should be more similar to } x_j \text{ then to } x_k\} \tag{2.4}$$

Suppose that we have the similar and dissimilar pairs of instances $(\mathbf{x}_i, \mathbf{x}_j)$ in the set $\mathcal{S}$ and $\mathcal{D}$ respectively. We would like to keep the distance of the similar pairs small, i.e. $d_A(\mathbf{x}_i, \mathbf{x}_j) \leq u$ when $(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{S}$, while separate the dissimilar pairs as possible, i.e. $d_A(\mathbf{x}_i, \mathbf{x}_j) \geq l$ when $(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{D}$, where u and l are constant.

$$\mathcal{S} = \{(x_{i,}x_j) \ : \ d_A(x_i, x_j) \leq u\} \tag{2.5}$$
$$\mathcal{D} = \{(x_i, x_j) \ : \ d_A(x_i, x_j) \geq l\} \tag{2.6}$$

For triplet-based constraints, we hope that the distance between dissimilar pairs should be greater than the distance between similar pairs with a fixed constant margin α.

$$\mathcal{R} = \{(x_i, x_j, x_k) \ : \ f_M(x_i, \ x_k) - f_M(x_i, x_j) \geq \alpha\} \tag{2.7}$$

A distance learning algorithm aims at finding the parameters of the distance function such that is best agrees with these constraints, in an effort to approximate the underlying semantic metric. This is typically formulated as an optimization problem [6] that has the following general form.

$$\min_{M} \quad L(M, \mathcal{S}, \mathcal{D}, \mathcal{R}) + \lambda R(M) \tag{2.8}$$

where $L(M, \mathcal{S}, \mathcal{D}, \mathcal{R})$ is a loss function that incurs a penalty when training constraints are involved. $R(M)$ is some regularizer on the parameters M of the learned distance function and $\lambda \geq 0$ is the regularization parameter.

The state-of-art distance learning algorithms share the same framework but formulate different formulas by their choice of metric, constraints, loss function, and regularizer.

After the distance learning phase, the resulting function is employed to improve the performance of a distance-based algorithm, which is most often k-Nearest Neighbors (k-NN), but may also be a clustering algorithm such as K-Means, a ranking algorithm. Recent years, distance learning is also able of boosting the performance of deep learning.

The form of a metric is the crucial property of distance learning algorithms. It usually has categories into three classes:

Global distance function: A single linear or non-linear distance function, such as the Mahalanobis distance and the

$\mathcal{X}^2$ histogram distance respectively. For linear distance function, their expression power is limited, but they are easier to optimize and less prone to over-fitting. Linear distance function usually lead to convex formulations, and thus global optimality of the solution is reachable. For non-linear distance function, they often give rise to non-convex formulations and may overfit, but they can capture non-linear variations in the data, which is common in big data.

Local distance function: Local distance metrics, where multiple (linear or non-linear) local metrics are learned to formulate complex problems, such as heterogeneous data with local structures. They are however easier to be over-fitting than global methods since the number of parameters they learn can be very large.

Deep distance function: distance learning functions are formulated as deep Siamese networks to learn a semantic representation of the input data by distance comparisons. Like deep learning framework, deep distance learning is difficult to train with conventional optimization techniques due to the non-convexity of the problems.

Distance learning has been a hot topic of research in the machine learning community for two decades and now achieves a considerable level of maturity in terms of theoretical formulation and practical issues. Except for a few early methods, most distance learning algorithms are competitive to achieve state-of-the-art performance on several problems like information retrieval and ranking. However, each algorithm has its intrinsic properties (e.g., type of metric, ability to leverage unsupervised data, good scalability with dimensionality, generalization guarantees) and emphasis should be placed on those when deciding which method to apply to a given problem.

The (squared) Mahalanobis distance, an extension of Euclidean distance, measures the distance between two points lie on the special linear space. It is defined as

$$d_{\mathbf{M}}(p_1, p_2) = (p_1 - p_2)^T \mathbf{M} (p_1 - p_2), \tag{2.9}$$

where $p_1, p_2 \in \mathbb{R}^d$ are input examples, $\mathbf{M}$ is a symmetric and positive semi-definite $d \times d$ matrix. When $\mathbf{M} = \mathbf{I}$, the Mahalanobis distance is equivalent to the Euclidean distance.

Since Xing et al, [128] formulate the distance function learning problem as an optimization problem and solve it by semi-definite programming, there are many distance learning algorithms in the literature, which are developed for different objectives, such as learning global distance function [102], local distance function [36] or any special structured distance function [77]. Completed surveys from different perspectives can be found in [7, 64]. Some algorithms focus on pairs-wise constraints [129], some solves the optimization problem with relative constraints [124]. Due to the high complexity of computation, the majority of DML algorithms target low-dimensional or intermediate dimensional data up to hundreds of dimensions.

Several algorithms [90, 93] attempt to meet the challenge of high-dimensional distance learning. [127] assumes the Mahalanobis matrix as low-rank matrix and represented as $\mathbf{R}\mathbf{R}^T$. Meanwhile, [90] takes another form of low-rank assumption that the Mahalanobis matrix is sparse. [132] reduces the computational complexity during the eigen-decomposition by picking up the largest eigenvalue and the corresponding eigen-

vector in a variant Frank-Wolfe algorithm [37]. [23] targets to the similar objective by taking the top-k eigenvalues and those corresponding eigen-vectors. [93] simplifies the high-dimensional distance learning with multiple stages. [81] reformulates the high-dimensional Mahalanobis matrix as a combination of low-rank matrices, which are computed from gradient boosting as weak learners. In this chapter, we also employ the similar formulation but with different approach to update the low-rank matrices.
There are plenty of algorithms aiming at learning such distance function by solving a semi-definite or a quadratic program [124, 128, 102]. Almost all the distance learning algorithms try to constrain similar data points and to scatter those dissimilar data points. Early work like [128] formulates this problem as an optimization problem on the second-order cone, which is costly solvable. Davis et al. [29], Weinberger et al. [124] and Shen et al. [102] formulate different kinds of optimization problems, namely ITML, LMNN, BoostMetric respectively. However, the common issue that their solutions are computationally expensive. Very recently, Zadeh et al. [135] propose a new objective function and give the closed-form solution from the geometric domain. It is the most promising global distance learning method because of the computational speed several orders of magnitude faster than the widely used ITML and LMNN methods.

2.2 Deep Distance Learning Framework

Siamese neural networks [11, 62] employ pair or triplet neural network with shared parameters to learn a non-linear function

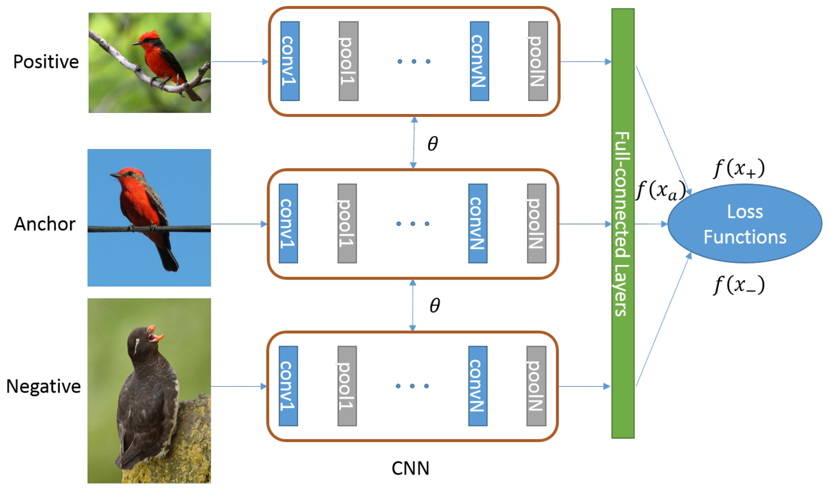


Figure 2.1: Siamese network. In triplet framework, the Siamese network generates the embedding data $f(x_a)$, $f(x_+)$, $f(x_-)$ for the anchor x_a, the positive x_+ and the negative x_- respectively.

embedding raw high-dimensional data into low-dimensional metric, in which a contrastive loss is trained to distinguish between similar and dissimilar pairs of data.

Let $x \in \mathcal{X}$ be an input data and $y \in \{1, \ldots, L\}$ is the corresponding output label. In the pair-wise framework, Siamese network is trained with contrastive loss calculated with pairs of examples (x_i, x_j) which are either similar or dissimilar. The objective of learning is that the distances between embedding vectors of similar examples with some fixed margin should be smaller than the distances between that of dissimilar examples. More precisely, it minimizes the following loss:

$$\begin{aligned}\ell(\mathcal{X}, Y) &= \frac{1}{|\mathcal{P}|} \sum_{(x_i,x_j)\in\mathcal{P}} y_{i,j} g(x_i, x_j) \\ &\quad + (1 - y_{i,j}) \left[\alpha - g(x_i, x_j)\right]_+ , \end{aligned} \tag{2.10}$$

where $g(\cdot,\cdot)$ represents the output of Siamese network for a given pair of input. The label $y_{i,j} \in \{0, 1\}$ indicates whether a pair (x_i, x_j) comes from the same class or not. The operation $[\cdot]_+$ denotes the hinge function which takes the positive component. α denotes a fixed margin. Usually, g is defined as a Euclidean distance between two embedding data points:

$$g\left(x_i, x_j\right) = \left\| f\left(x_i\right) - f\left(x_j\right) \right\|_2 , \tag{2.11}$$

where $f(x)$ generates the representing features of the given input x in deep neural network.

The main idea is to construct a Euclidean space to make sure that positive pairs are close to each other while negative pairs are pushed away. Although the training process only requires a weaker form of supervision, the contrastive loss is focused on absolute distances, where the relative distance is more critical in many situations.

Similarly, in triplet-wise framework [124], the model is trained with triplets of examples (x_a, x_+, x_-). The positive and negative data of a given anchor point x_a is denoted as x_+ and x_- respectively. It means that x_a and x_+ comes from the same class while x_- is from different class to x_a. The objective of learning is that the distance between embedding vectors of similar pair (x_a, x_+) is less than that of dissimilar pair (x_a, x_-). Within the triplet-wise framework, [99] constructs triplets by finding a semi-hard negative data, which has the smallest distance to the anchor point x_a among all negative data points $\{x_-\}$.

The triplet loss is defined as the following:

$$\ell\left(\mathcal{X}, Y\right) = \frac{1}{|\mathcal{T}|} \sum_{(x_a, x_+, x_-) \in \mathcal{T}} \left[g\left(x_{a,} x_+\right) + \alpha - g\left(x_a, x_-\right) \right]_+ , \tag{2.12}$$

where $\mathcal{T}$ is the set of triplets.

However, the performance of deep distance learning algorithms relies on the quality of positive and negative data pairs. Usually,

carefully-designed negative data mining is expensive for deep network because there is no informative negative supervision during the early stage of the training process.
[104] propose several valuable N-pair batch construction within less computational burden:

- $(N+1)$-tuplet loss from N negative samples:

$$\mathcal{L}\left(\left\{x_i, x_i^+\right\}_{i=1}^N ; f\right) = \tag{2.13}$$
$$\log\left(1+\sum_{i=1}^N \exp\left(f^\top f_i^+ - f^\top f_i^+\right)\right) \tag{2.14}$$

- Multi-class N-pair loss:

$$\mathcal{L}_{N-pair-mc}\left(\left\{x_i, x_i^+\right\}_{i=1}^N ; f\right) = \tag{2.15}$$
$$\frac{1}{N}\sum_{i=1}^N \log\left(1+\sum_{j\neq i} \exp\left(f_i^\top f_j^+ - f_i^\top f_i^+\right)\right) \tag{2.16}$$

- One-vs-one N-pair loss:

$$\mathcal{L}_{N-pair-ovo}\left(\left\{x_i, x_i^+\right\}_{i=1}^N ; f\right) = \tag{2.17}$$
$$\frac{1}{N}\sum_{i=1}^N \sum_{j\neq i} \log\left(1+\exp\left(f_i^\top f_j^+ - f_i^\top f_i^+\right)\right) \tag{2.18}$$

[106] considers all the positive and negative pairs, they follow [99] to randomly sample several positive pairs, then select their semi-hard negative pairs to the training mini-batch.
Finally, they optimize a structured prediction objective on the lifted problem by converting the vector of pair-wise distances in the mini-batch to the matrix of pair-wise distance. [103] finds

an aggressive mining method to improves the discrimination of model by combing both positive and negative pairs during the training. [105] finds that minimizing the distance loss for mini-batches does not necessarily lead to a more discriminative distance function and propose to consider the global structure of the distance function during the training.

The following two ideas closely relate to deep distance learning, but it is difficult to extend these methods into a distributed setting. [68] proposes an effective deep distance learning approach by carefully partitioning the training dataset. [114] estimates the distribution of positive and negative data pairs and takes advantages of a distribution loss for deep distance learning.

Although there is no existing distributed framework designed for the deep distance learning algorithm, we can easily employ conventional distributed deep learning frameworks [86, 137] to boost deep distance learning algorithms. Parameter server [72] is a widely adopted solution in distributed deep learning platforms [1, 24] The challenge is how to allocate resources between workers and parameter servers to fully utilize CPU and reduce the communication workload around the parameter servers. [122, 75] explore to reduce the communication demand in deep learning framework by decentralizing the process of the gradient aggregation. However, for many deep distance learning algorithms, a straightforward adoption from a single machine to distributed environment cannot achieve a remarkable performance gain. We will discuss the reasons with more details in chapter 5.

2.3 Distance Learning between Probability Distributions

In optimal transport theory, we employ probability simplex defined in Eq. (eq:distribution-probability-simplex) to represent arbitrary probability distributions.

$$\Sigma_n = \left\{ p \in \mathbb{R}_+^n \quad : \quad \sum_{i=1}^{n} p_i = 1 \right\} \tag{2.19}$$

A discrete empirical distribution with weight p and locations $\{x_1, \ldots, x_n\}$ is defined as:

$$\mu = \sum_{i=1}^{n} p_i \delta_{x_i} \tag{2.20}$$

When we consider two empirical distribution μ_a and μ_b, the transport map from μ_a to μ_b is defined as:

$$\Pi\left(\mu_a, \mu_b\right) = \left\{ \pi \in \mathbb{R}_+^{n_a \times n_b} \mid \pi \mathbf{1}_{n_b} = \mu_a, \pi^T \mathbf{1}_{n_a} = \mu_b \right\} \tag{2.21}$$

where δ_{x_i} is the Dirac function at location $x_i \in \mathbb{R}^d$. p^a and p^b are probability simplex.
In the constraint in Eq. (2.21), the π is a sparse matrix with at most $n_a + n_b - 1$ non-zero elements, equating the rank of the constraint matrices.
Derived from the optimal transport problem, the p-Wasserstein distance between μ and ν is defined as

$$W_p\left(\mu, \nu\right) = \left(\inf_{\gamma \in \Pi(\mu, \nu)} \int \int_{\mathcal{X} \times \mathcal{X}} d\left(x, y\right)^p \mathrm{d}\gamma\left(x, y\right) \right)^{1/p} \tag{2.22}$$

where $\Pi\left(\mu, v\right)$ is the set of all joint distributions. $\gamma\left(x, y\right)$ relates how much "mass" must be transported from x to y in order to transform the distribution μ to the distribution ν. The infimum in Eq. (2.22) is highly intractable.
When $p = 1$, W_1 is also known as Earth Mover's distance or Monge-Kantorovich distance.

$$W\left(\mu, \nu\right) = \inf_{P \in \Pi(\mu,\nu)} \mathbb{E}_{(x,y)\sim P}\left[\|x - y\|\right] \tag{2.23}$$

Kantorovich-Rubinstein duality in Eq. (2.24) is employed to solve this problem.

$$W\left(\mu, v\right) = \sup_{\|f\|_L \leq 1} \mathbb{E}_{x\sim\mu}\left[f\left(x\right)\right] - \mathbb{E}_{x\sim v}\left[f\left(x\right)\right] \tag{2.24}$$

where the supremum is over all the 1-Lipschitz functions.
The Kantorovich-Rubinstein duality with discrete distributions is defined as:

$$W_p^p\left(\mu, \nu\right) = \max_{\substack{\alpha \in \mathbb{R}^n, \beta \in \mathbb{R}^m \\ \alpha_i + \beta_j \leq d\left(x_i, y_j\right)^p}} \alpha^T a + \beta^T b \tag{2.25}$$

When the points are fixed, the natural choice for the weights is $p_i^a = \mu_a\left(V_i\right)$ and $p_j^b = \mu_b\left(W_j\right)$. Where V_i and W_j represent the Voronoi cells of the point x_i^a and y_j^b respectively.
Given a finite family of distinct points $(x_i)_i \subset \mathbb{R}^d$, the Voronoi cells are defined as follows:

$$V_i = \{x \in \mathbb{R}^d : |x - x_i| \leq |x - x_j| \forall j\} \tag{2.26}$$

The most promising application about Wasserstein distance between probability distributions is Generative Adversarial Networks (GAN). The GAN defines two competing networks: The generator network G produces a data x from a source of noise $\mathbf{z} \sim \mathbb{P}_z$. The discriminator network D is trained to distinguish between the generated fake sample and a real sample. Formally, the GAN defines the following minimax objective:

$$\min_G \max_D \mathbb{E}_{x_r \sim \mathbb{P}_r} \left[\log \left(D \left(x_r\right)\right)\right] + \mathbb{E}_{x_g \sim \mathbb{P}_g} \left[\log \left(1 - D \left(x_g\right)\right)\right], \tag{2.27}$$

where $\mathbb{P}_r$ and $\mathbb{P}_g$ denote the real distribution and the generated distribution respectively. Practically, the expectations are empirically evaluated using samples. Unfortunately, well-learned discriminators may suppress the training of generators. We need to well tune the rounds of discriminator updates after every generator update to ease this training instability.

To enhance the training stability, [4] introduces Wasserstein-1 distance to GAN framework. The Wasserstein distance is made known as a more geometric-aware cost function for learning the distributions supported by the low-dimensional manifold, which is a widely adopted assumption in the feed-forward neural network.

Wasserstein GAN (WGAN) employs Kantorovich-Rubinstein duality of Wasserstein-1 distance, which relies on 1-Lipschitz continuity of the discriminator:

$$W_1 \left(\mathbb{P}_r, \mathbb{P}_g\right) = \sup_{\left\|D_\theta\right\|_L \leq 1} \mathbb{E}_{x_r \sim \mathbb{P}_r} \left[D_\theta \left(x_r\right)\right] - \mathbb{E}_{x_g \sim \mathbb{P}_g} \left[D_\theta \left(x_g\right)\right], \tag{2.28}$$

where $\mathbb{P}_g \sim G_\vartheta \left(\mathbb{P}_z\right)$. To approximate the supremum in Eq. (2.28), [4] proposes to clip the parameters θ to enforce the discriminator D_θ to be 1-Lipschitz.

Currently, the Wasserstein GAN approach is considered as the state-of-the-art method due to the theoretical contributions and competitive performance. However, to approximate the 1-Lipschitz constraint is very challenging in the Kantorovich-Rubinstein dual form of the Wasserstein-1 distance function. [48] introduces a soft penalty for the violation of 1-Lipschitzness (WGAN-GP). The gradient is evaluated as a linear interpolation between the training data and generated samples as a proxy to the optimal coupling. The gradient penalty only takes effect on the observed data $\{\mathbf{x}\}$, the other support domain is not covered. [123] follows WGAN-GP to enforces the Lipschitz continuity over all the data manifold and its surrounding regions.

An alternative way to solve the problem in Eq. (2.22) is to minimize the primal of optimal transport. The primal formulation is numerically stable because it does not involve differentiating the dual solution. [8] proposes to minimize a regularized primal form of optimal transport problem. [89] explores the theoretical properties of such regularization under Wasserstein GANs framework. [33] employs random projection to approximate the Wasserstein distance directly. The discriminator is not mandatory in their approach. [44] proposes a divergence measurement based on the Sinkhorn algorithm, which is originally designed for discrete optimal transport with entropic regularization.

To tackle with parallel computation, [20] proposes an ensemble method of pairs of the generator and discriminator, which can be trained in parallel naturally. However, the communication demand is severe, because the “messenger” discriminators and generators with a massive number of parameters need to be synchronized among workers.

Chapter 3

Semi-synchronous Algorithm for Global Distance Learning

Distance learning is an effective similarity learning tool to learn a distance function from examples to enhance the model performance in applications of classification, regression, and ranking and so on. Most distance learning algorithms need to learn a Mahalanobis matrix, a positive semi-definite matrix that scales quadratically with the number of dimensions of input data. This brings the huge computational cost in the learning procedure, and makes all proposed algorithms infeasible for extremely high-dimensional data even with the low-rank approximation. Differently, in this thesis, we first present the primitive knowledge of distance learning and then demonstrate a distributed solution of a viral distance learning algorithm called Information-Theoretic Metric Learning (ITML). More specifically, we utilize the property that each positive semi-definite matrix can be decomposed into a combination of rank-one and trace-one matrices and convert the original sequential training procedure into a parallel one. In most cases, the communication demands of the proposed method are also reduced from $\mathcal{O}(d^2)$ to $\mathcal{O}(cd)$, where d is the number of dimensions of the data and c is the number of constraints in distance learning and can be smaller than d by appropriate selection. Moreover importantly, we present a rigorous theoretical analysis to upper bound the Bregman divergence between the sequential algorithm and the parallel algorithm, which guarantees the correctness and performance of the proposed algorithm.

3.1 Problem and Motivation

Learning an appropriate distance function is an important topic in both machine learning and data mining. Distance learning has been widely applied in many problems, such as image retrieval [65], face recognition [65], bioinformatics analysis [61], software error detection [29, 64]. Earlier publications formulate the learning problem as a convex optimization problem by maximizing the sum of the difference between the dissimilar instances while restricting the distance between the similar instances to be small [128]. However, solving such problem by semi-definite programming (SDP) solver or eigenvalue decomposition is time-consuming and poorly scalable to medium and high dimensional data. Recent developments such as Large Margin Nearest Neighbors (LMNN) [32, 124], BoostMetric [102] and Information-Theoretic Metric Learning (ITML) [29] try to speedup distance learning by the special structure of Mahalanobis matrix, partial eigenvalue decomposition or low-rank approximation.

However, Most of the existing algorithms are incapable of handling high-dimensional data. LMNN is very easy to overfit for high-dimensional data. BoostMetric employing the max eigenvalue decomposition needs a huge number of iterations to converge for high-dimensional data. ITML conducts heavy matrix multiplication which is very time-consuming for high-dimensional data.

For other earlier work, is a popular solution for distance learning problem but it is very easy to overfit especially for high-dimensional data. [100] relies on the special structure of Mahalanobis matrix. [102] is an efficient distance learning solution only related with the max eigenvalue decomposition, however, it needs a huge number of iterations to converge for high-dimensional data.

In the high dimensional setting, most existing work mainly exploit the low-rank structure of the learned matrix or the sparsity of the covariance matrix. Instead of learning the full rank matrix $\mathbf{A}$, [28] factorizes the matrix $\mathbf{A}$ as $\mathbf{R}\mathbf{R}^T$ and learned a low-rank approximation of the Mahalanobis matrix. [90] imposes a different prior on learning the Mahalanobis distance based on the sparsity of sample concentration matrix Σ^{-1}, where Σ is the covariance matrix of samples. [45] proposed the low dimensional projection to extract a compact feature representation for originally high dimensional data. However, all the above properties may not hold for highly complex data in real-life applications. Although each of these algorithms was shown to yield excellent performance in classification and clustering tasks, they do not generalize learning an arbitrary metric, parameterized by an arbitrary matrix. To the best of our knowledge, it is still nontrivial to learn the approximated distance between two instances with high dimensionality if the low-rank property is not satisfied. Hence, how to learn the distance without highly depending on the low-rank assumption still requires sophisticated designs and experimental explorations.

Meanwhile, with large volumes of data available in many real life applications, the problem of scalability becomes the key concern in designing many machine learning algorithms. Most of the existing work in scalable machine learning relies on approximation methods and parallel computing schemes. Specifically, in term of the algorithm aspect, most existing work mainly focus on stochastic gradient descent method for convex optimization problem [2, 113, 141] or proximal descent method for non-convex problem [22, 73, 107]. Few efforts have been made on solving the semi-definite programming problem for matrices in a distributed cluster, especially for distance learning problem. Very recently, [71] proposed a distributed approach to speedup the computation of distance learning problem. Unfortunately, both of its methodology and proofs heavily rely on the low-rank property of Mahalanobis matrix. Forcing low-rank assumption without understanding the data may destroy the completeness of the data and yield sub-optimal performance.

To address the above two challenges, taking advantage of powerful multi-thread cluster, we will explore to design a general parallel approach to solve distance learning problem for high-dimensional data without low-rank assumption about the Mahalanobis matrix. In this chapter, we will design a scalable solution to speedup the training of ITML, which repeats Bregman projection [10]. By decomposing the Mahalanobis matrix into a linear combination of rank-one matrices, and dispatching them into different machines like a divide-and-conquer scheme, our implementation performs Bregman projection on these rank-one matrices without low-rank assumption about the Mahalanobis matrix. Finally, after iterations, the Mahalanobis matrix can be reconstructed from these rank-one matrices to calculate the pair-wise distances.

In summary, the proposed work contributes on the following aspects:

- By making a trade-off between accuracy and running time, we first propose a scalable approach to learn distance functions from high-dimensional data without low-rank assumption.
- We compare the proposed method with the original ITML method from a theoretical perspective and give an upper error bound brought by our parallel update.
- The experiments conducted on the data with different scales demonstrate the correctness and the scalability of our method.

3.2 Methodology

3.2.1 Information-Theoretic Metric Learning

We use the same formulation introduced in section 2.1 of Chapter 1.
ITML is one of the start-of-art algorithm for distance learning. It brings an assumption that the data come from multivariate Gaussian distribution, which is very common in most applications. Under this assumption, the Mahalanobis distance matrix A can be expressed as the inverse of the covariance matrix of the multivariate Gaussian distribution. Because the probability density function of multivariate Gaussian can be reformulated as $p(x; A) = \frac{1}{Z}\exp(-\frac{1}{2}d_A(x, \mu))$. Hence, the original DML problem is expressed as finding a proper multivariate Gaussian by minimizing difference relative entropy with the initial Mahalanobis distance function. The objective function is reformulated as:

$$\begin{aligned} \min \quad & KL(p(x;A_0)||p(x;A) \\ s.t. \quad & d_A(x_i,x_j) \leq u \quad (i,j) \in S \\ & d_A(x_i,x_j) \geq l \quad (i,j) \in D \end{aligned}$$

where,

$$\begin{aligned} & KL(p(x|\mu_0,A_0)||p(x|\mu,A) \\ = & \int p(x|\mu_0,A_0)\log\frac{p(x|\mu_0,A_0)}{p(x|\mu,A)}dx \end{aligned} \tag{3.1}$$

A_0 is usually assigned to the identity matrix.
More specifically, the KL divergence in Eq. (3.1) can be expressed as a convex combination between a Mahalanobis distance between means and the Bregman divergence with respect to function $\phi(A) = -\log\det(A)$ between covariance matrices [34]:

$$KL(p(x|\mu_0,A_0)||p(x|\mu,A) = \frac{1}{2}D_\phi(A,A_0) + \frac{1}{2}d_{\Sigma^{-1}}(\mu_0,\mu) \tag{3.2}$$

where the Bregman divergence is defined as:

$$D_\phi(A,A_0) = \phi(A) - \phi(A_0) - \operatorname{tr}\left(\nabla\phi(A_0)^T(A-A_0)\right) \tag{3.3}$$

With the assumption that the mean vectors of two Gaussian distribution are the same, ITML formulates the DML problem as:

$$\begin{aligned} \min \quad & D_\phi(\mathbf{A},\mathbf{A}_0) \\ s.t. \quad & \operatorname{tr}(\mathbf{A}(\mathbf{x}_i-\mathbf{x}_j)(\mathbf{x}_i-\mathbf{x}_j)^T) \leq u, \quad (\mathbf{x}_i,\mathbf{x}_j) \in \mathcal{S} \\ & \operatorname{tr}(\mathbf{A}(\mathbf{x}_i-\mathbf{x}_j)(\mathbf{x}_i-\mathbf{x}_j)^T) \geq l, \quad (\mathbf{x}_i,\mathbf{x}_j) \in \mathcal{D} \end{aligned} \tag{3.4}$$

This problem is then cast into a particular Bregman Matrix Divergence problem [66], which contains global optimal solution and can be solved efficiently by Eq. (3.5) without eigenvalue computations or semi-definite programming.

$$A_{t+1} = A_t + \beta A_t(\mathbf{x}_i - \mathbf{x}_j)(\mathbf{x}_i - \mathbf{x}_j)^T A_t \tag{3.5}$$

where β is a scalar for the learning rate. The actual distance between a pair of data sample (x_i, x_j) in each iteration is computed from:

$$p(t) = (x_i - x_j)^T A_t (x_i - x_j) \tag{3.6}$$

3.2.2 Distributed Distance Learning

The algorithm (see details in [29]) repeats the Bregman projection in Eq. (3.5) with different constraints sequentially. For each projection, it involves the computation on $A_t \in \mathbb{R}^{d\times d}$, where d is the number of dimensions of the input data. When d is large enough, the running time of the Bregman projection in a single PC is not acceptable. That why the previous work with sequential update is only capable of handling the data with $\mathcal{O}(10^2)$-dimension.

In this chapter, we try to conduct the Bregman projection with the power of parallel computation. The key step in Eq. (3.5) is to update the Mahalanobis matrix which is a $d \times d$ positive semi-definite matrix. Typically, there are two roads to parallel this update: 1) compute the matrix multiplication in Eq. (3.5) directly in a distributed way. For example, the matrix is decomposed into blocks which are dispatched into different machines. When the local computation with respect to blocks finishes, the intermediate results are collected to form a complete matrix update. The drawback of this kind of approaches is obvious that there is heavy communications demand for carrying out the data in blocks; 2) design a Hogwild! [95]-like approach to update the Mahalanobis matrix based on partial information simultaneously. Unfortunately, the positive semi-definite property of Mahalanobis matrix is difficult to hold from multiple partial updates.

In order to hold the positive semi-definite property, we borrow the idea from [102] that any positive semi-definite matrix can be decomposed into a linear combination of rank-one matrices. Therefore, we have proposed a novel parallel solution for DML problem by reformulating the Mahalanobis matrix A as $A = I + \sum_i \alpha_i z_i z_i^T$ where $z_i \in \mathbb{R}^d$. By observing the update in Eq. (3.5), it is easy to find that $A_t(x_i - x_j)$ is a vector with d dimension and $A_t(x_i - x_j)(x_i - x_j)^T A^T$ is a rank-one matrix. Hence, we can concrete the formula of A by summing over the Bregman projection through all pairs of constraints in Eq. (3.7)

$$A_{t+1} = I + \sum_{i=1}^{C} \beta_i(t) z_i(t) z_i^T(t) \tag{3.7}$$

where, I is the unit matrix with the size of $d \times d$, C is the number of pairs of constraints, β is the step from Bregman projection in Eq. (3.5) and $z_i(t) = A_t c_i$. $c_i = x_j - x_k$ for the i-th pair of constraint (x_j, x_k). Actually, in the proposed framework, we only store z rather than A_t because A_t is a huge matrix when d becomes large. Consequently, the update of z is changed to Eq. (3.8)

$$\begin{aligned} z_k(t+1) &= A_{t+1}c_k \\ &= \left(I + \sum_{i=1}^{C} \beta_i(t) z_i(t) z_i^T(t)\right) c_k \end{aligned} \tag{3.8}$$

Based on the original algorithm in [29], $\beta_i(t)$ is a function of $p_i(t)$ in Eq. (3.6) and the upper bound or lower bound of constraints (see the details in Algorithm 1). The key step of updating $\beta_k(t)$ is to compute the actual distance $p_k(t)$. Combined with Eq. (3.7), the actual distance for the k-th constraint c_k can be expressed in Eq. (3.9)

$$\begin{aligned} p_k(t) &= c_k^T A_t c_k \\ &= c_k^T \left(I + \sum_{i=1}^{C} \beta_i(t) z_i(t) z_i^T(t)\right) c_k \\ &= c_k^T c_k + \sum_{i=1}^{C} \beta_i(t) c_k^T z_i(t) z_i^T(t) c_k \end{aligned} \tag{3.9}$$

Due to the separable property of the Mahalanobis matrix A in Eq. (3.7), we can dispatch the tasks for updating z in Eq. (3.8) and p in Eq. (3.9) into C workers which is an abstract concept representing the parallel executors, such as process in a single machine or different machines.

In our framework, worker k needs to receive all the update of z from other workers at the previous iteration first, then computes the update for the next iteration.

Overall, each work sends only one vector z rather than the matrix $\beta A(x_i - x_j)(x_i - x_j)^T A$ and receive $(C-1)$ times z rather than the whole Mahalanobis matrix in the original sequential algorithm. Hence, the communications demand for each work reduces from $\mathcal{O}(d^2)$ to $\mathcal{O}(cd)$. When the number of dimensions

exceeds the number of constraints, which is a very commonly happened situation for high-dimensional distance learning tasks, the demand of communications reduces significantly compared with the original ITML algorithm.
Furthermore, when the number of constraints keeps relative small, our work is equivalent to an ITML-based method with low-rank assumption about the Mahalanobis matrix. If the number of constraints becomes large, even larger than the number of dimensions, our method still works although the cost for communications will increase inevitably. Therefore, our method is a general parallel approach no matter the Mahalanobis matrix is low-rank or not.

3.2.3 Implementation

We implement our method on Apache Spark [136], which is a general propose, high-efficient, distributed in-memory computing platform. It provides a flexible data structure called Resilient Distributed Dataset (RDD) to support parallel computing on RDDs. The data or parameters in RDDs are cached in memory instead of loading from disk at each iteration like Hadoop [1]. Usually, the content of RDD is distributed in different machines and will be re-dispatched automatically without manual operation. Therefore, Spark is suitable for developers who only need to provide high-level logic to parallel the algorithm instead of caring about many details in a cluster. In our implementation, the list of z and the corresponding β are stored in an RDD. For each element of this RDD, we assign a "mapper" function to compute the update of z in Eq. (3.8) and the partial distance in Eq. (3.9) . All these "mapper" functions

[1]https://hadoop.apache.org/

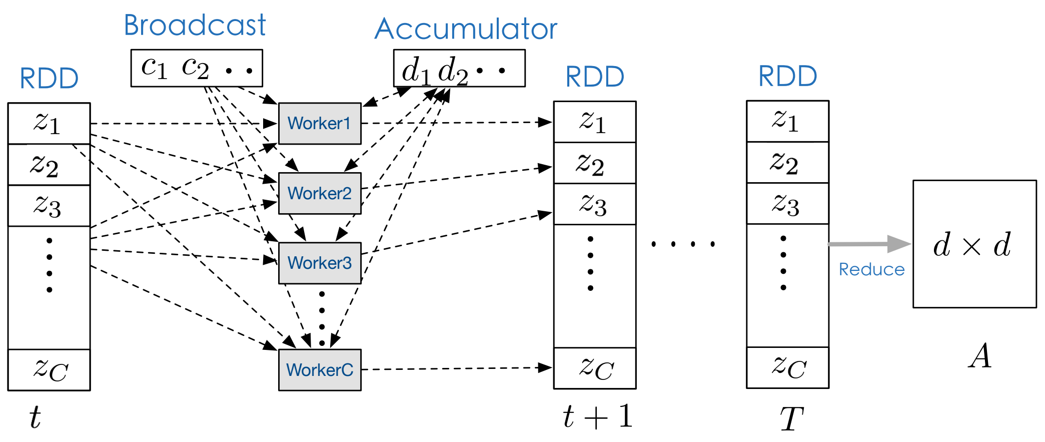


Figure 3.1: Framework of parallel distance learning on Apache Spark

will generate a new RDD to store the new z for next iteration. We also declare a constant "Broadcast" variable which is shared for all workers to store constraints c, because the computation of new z in each worker needs its corresponding c and the computation of partial distance iterates all constraints. In most cases, the "Broadcast" variable is cached in memory and can be fetched efficiently. Finally, another special component in Spark, called "Accumulator" will be created to collect the partial distance from each worker.

With the help of the global "Broadcast" and "Accumulator", the whole distributed algorithm is conducted in bulk synchronous parallel (BSP) [9], which guarantees that all parameters will be updated within the same iteration. More specifically, the employment of "Broadcast" and "Accumulator" in our implementation illustrated in Fig. 3.1 significantly reduce the redundant computation of the distance between each pair of data sample under the learned metric space. Because the procedure of computing distance is divided into blocks which are shared among all workers.

3.3 Theoretical Analysis

In this section, we will analyze the difference and give the upper bound of the proposed algorithm compared with the sequential algorithm in [29]. Because the result of the parallel computation of z in our algorithm may be different with the sequential version of the original algorithm.

First, we will compare the difference between the sequential version and the parallel version. In sequential order, the algorithm conducts C times Bregman projection to cover all pairs of constraints. After C iterations since iteration t, the Mahalanobis matrix A_{t+C} can be expressed as:

$$A_{t+C} = A_t + \sum_{i=1}^{C} \beta_i A_{t+i-1} c_i c_i^T A_{t+i-1}^T \tag{3.10}$$

For parallel version algorithm, if we rearrange the order of computation like a series of sequential updates, one iteration in the parallel version algorithm is equal to C iterations in a sequential algorithm but with the same A_t rather than A_{t+i-1}. The formula of parallel algorithm is expressed in Eq. (3.11)

$$\hat{A}_{t+C} = A_t + \sum_{i=1}^{C} \beta_i A_t c_i c_i^T A_t^T \tag{3.11}$$

By observing the difference between Eq. (3.10) and Eq. (3.11), we can find that the Bregman projection related with the i-th constraint in parallel algorithm is applied on a i-delayed Mahalanobis matrix. Figure 3.2 illustrates the difference between the normal update and the delayed update.

For convenience, we use the sequential version with delayed update to represent the parallel algorithm.

Theorem 1 *Delayed Update under Bregman Divergence*

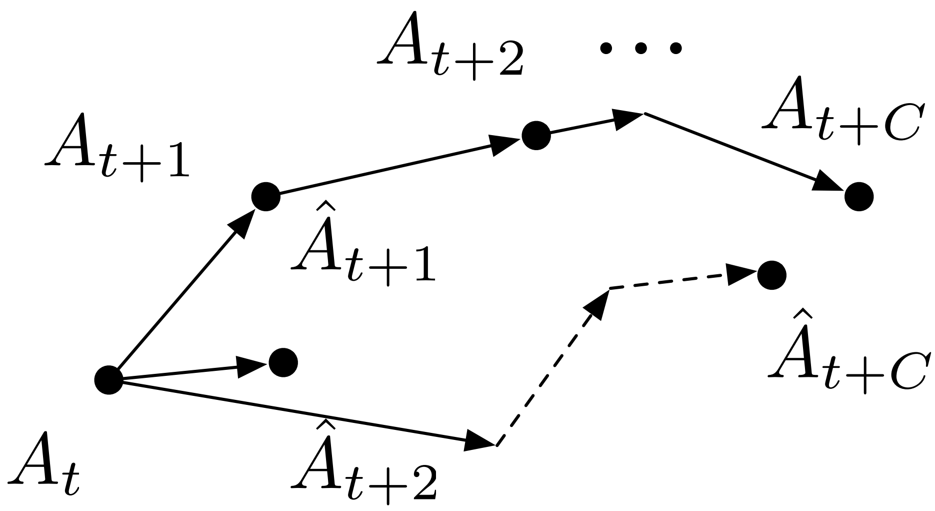


Figure 3.2: Delayed vs. normal versions of projection

Assume the difference of matrices is measured by the Bregman divergence with respect to LogDet divergence $\phi(X) = -\log\det(X)$. The minimizer of $D_\phi(A_0, A)$ after T iterations is A^*. The length of convergence path is denoted as Ω. The upper error bound of Algorithm 1 is

$$R[A] := \sum_{t=1}^{T} D_\phi(A_t, A^*) \leq \frac{1}{\beta_{\min}} D_\phi(A^*, I) + \frac{1}{2} L\Omega \tag{3.12}$$

Proof: From the definition of Bregman divergence and the convexity of function $\phi(A)$ we know:

$$\begin{aligned}
R[A] &= \sum_{t=1}^{T} D_\phi(A_t, A^*) \\
&= \sum_{t=1}^{T} \phi(A_t) - \phi(A^*) - \operatorname{tr}(\nabla\phi(A^*)^T(A_t - A^*) \\
&\leq \sum_{t=1}^{T} \phi(A_t) - \phi(A^*) \\
&\leq \sum_{t=1}^{T} \operatorname{tr}\left(\nabla\phi(A_t)^T(A_t - A^*)\right) \\
&= \sum_{t=\tau}^{T+\tau} \operatorname{tr}\left(\lambda_{t-\tau}^T(A_t - A^*)\right)
\end{aligned}$$

From Lemma 10 in [67], we have easily derived the matrix version as:

$$\begin{aligned}
\operatorname{tr}\left(\lambda_{t-\tau}^T\left(A_t - A^*\right)\right) &\leq \frac{1}{\beta_{t-\tau}}\left(D_\phi(A^*, A_t) - D_\phi(A^*, A_{t+1})\right) \\
&\quad + \beta_{t-\tau}\frac{||\lambda_{t-\tau}||_\star^2}{2} \\
&\leq \frac{1}{\beta_{\min}}\left(D_\phi(A^*, A_t) - D_\phi(A^*, A_{t+1})\right) \\
&\quad + \beta_{t-\tau}\frac{||\lambda_{t-\tau}||_\star^2}{2} \qquad (3.13)
\end{aligned}$$

where $||\cdot||_\star$ is the Fenchel-Legendre dual norm [101].
Sum up these inequality in Eq. (3.13) as :

$$
\begin{aligned}
&\sum_{t=\tau}^{T+\tau} \operatorname{tr}\left(\lambda_{t-\tau}^{T} A - A^{*}\right) \\
\leq &\frac{1}{\beta_{\min}} \sum_{t=\tau}^{T+\tau} \left(D_{\phi}(A^{*}, A_{t}) - D_{\phi}(A^{*}, A_{t+1})\right) \\
&+ \sum_{t=\tau}^{T+\tau} \beta_{t-\tau} \frac{||\lambda_{t-\tau}||_{\star}^{2}}{2} \\
= &\frac{1}{\beta_{\min}} \left(D_{\phi}(A^{*}, A_{t}) - D_{\phi}(A^{*}, A_{T})\right) + \sum_{t=\tau}^{T+\tau} \beta_{t-\tau} \frac{||\lambda_{t-\tau}||_{\star}^{2}}{2} \\
\leq &\frac{1}{\beta_{\min}} D_{\phi}(A^{*}, I) + \frac{1}{2} \sum_{t=0}^{T} \beta_{t} ||\lambda_{t}||_{\star}^{2} \\
\leq &\frac{1}{\beta_{\min}} D_{\phi}(A^{*}, I) + \frac{L}{2} \sum_{t=0}^{T} \beta_{t} ||\lambda_{t}||_{\star} \\
\leq &\frac{1}{\beta_{\min}} D_{\phi}(A^{*}, I) + \frac{1}{2} L\Omega
\end{aligned}
$$

When $t \leq \tau$, $A_t = A_0 = I$ and $L = \sum_{t=0}^{T} \beta_t ||\lambda_t||_{\star}$ is the length of the path of the Bregman projection in the original algorithm without any delayed update. Typically, L has a upper bound Ω as long as the original algorithm converges in the convex set.

3.4 Evaluations

In this section, we first demonstrate the scalability of our proposed DITML algorithm on Apache Spark by comparing it with 1) the original ITML algorithm and 2) its implemented in conventional distributed scheme on high-dimensional data from four synthetic datasets and the ImageNet dataset. The k-NN classifiers ($k = 1$ and $k = 5$) are employed to evaluate the accuracy of the learned distance metric space.

3.4.1 Experimental Settings

All implementations are written in Scala 2.10 and run on Apache Spark 1.6.0 with YARN as the cluster controller on 32 physical machines with 4TB memory and 668 executors in total. All 32 physical machines are inter-connected with 10 Gbps network switch. To make a fair comparison, we also implement the original ITML with the help of distributed matrix Scala classes in Spark 1.6.0.

3.4.2 Synthetic Datasets

We generate four sets of synthetic datasets on binary classification with dimensions in $10^{[2:1:5]}$ to test the scalability of our proposed framework with respect to the number of dimensions. More specifically, we employ the Scala object "KMeansDataGenerator" of "MLLib" package in Apache Spark to evenly generate the synthetic data following the normal distribution. The number of generated data for each dataset is one-tenth of the corresponding dimension. That is, the sizes of the datasets are $10, 100, 1000, 10000$, respectively.
For each dataset, we randomly sample pairs of data points from the same class to generate similarity constraints, where the upper bound u is assigned to the Euclidean distance between the pairs. The dissimilarity constraints are sampled from pairs of data points belonging to different classes, where the lower bound l is assigned to the Euclidean distance between the dissimilar pairs. Noted that, the number of similarity constraints is the same as the number of dissimilarity constraints, which is a common setup for distance learning. Hence, we collect $10, 100, 1000, 10000$ constraints, respectively, for these

four synthetic datasets.

3.4.3 ImageNet Dataset

For the ImageNet dataset, we randomly choose 50 pictures from two randomly selected classes, where each of them has 25 pictures as training data. DeCAF features [35], which capture most of the images information, are employed to represent the images in the experiment. The DeCAF features are extracted from a deep convolutional network following the procedure in [93]. We then obtain 51,456 DeCAF features for each image in the test.
To determine the upper bound u and the lower bound l, we use the median value of Euclidean distance as a threshold for all pairs of data following the standard setup in [7]. For the pair of data belong to the same class, if the Euclidean distance is greater than the median value, then we assign the Euclidean distance as the upper bound of the similar pair, else assign the median value as the upper bound of the similar pair. For the pair of data belong to different classes, the rule is similar to that for similar pairs. If the Euclidean distance is greater than the median value, then we assign the median value as the lower bound of the dissimilar pair, else we assign the Euclidean distance as the lower bound of the dissimilar pair. Overall, we have generated 1225 pair of constraints to train the Mahalanobis matrix in DML problem.

3.4.4 Scalability of Our Framework

Figure 3.4 shows the running time of our method with respect to the original ITML execution on the Spark cluster. We have the

following observations: 1) When the number of workers is less than 400, the total running time decreases significantly when the number of workers increases. 2) When the number of workers exceeds 400, the communication dominates the whole process and the speedup is not obvious when new workers come.
In order to display the advantage of the proposed DITML algorithm on the scalability issue compared with the original ITML, we implement the original ITML in Spark and transform the sequential execution of ITML to a distributed way by involving a distributed matrix class called “BlockMatrix”, which is a primitive data type in “MLLib” package in Apache Spark. Figure 3.3 displays direct comparisons of two kinds of ITML and the advantage of the proposed DITML algorithm compared with the original ITML on the execution time is significant. Overall fig. 3.3 and fig. 3.4 illustrate a good scalability of our method.

3.4.5 Performance of The Learned Metric Space

For the performance issue, Table 3.1 shows the accuracy of k-NN classification on the 4 synthetic datasets and the ImageNet. We use the Mahalanobis matrix learned from our method to measure the distance between two samples, which is used in k-NN classification. Compared with the standard k-NN method without changing the metric, the proposed method enhances the accuracy of k-NN classification from 0.682 to 0.83 on ImageNet dataset.
The comparison between ITML and the proposed method displayed in Fig. 3.5 shows both of the algorithms will converge with the similar number of iterations. In these experiments, the slack variable γ is assigned to 100. Table 3.1 also demonstrates the proposed method is able to improve the performance of k-NN

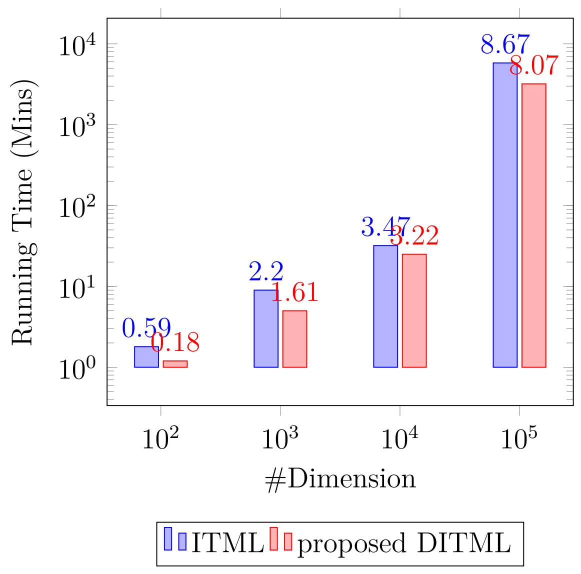


Figure 3.3: Comparisons between the original ITML and proposed DITML algorithms on different scaled synthetic datasets. Both of two algorithms iterates 50 times over all constraints.

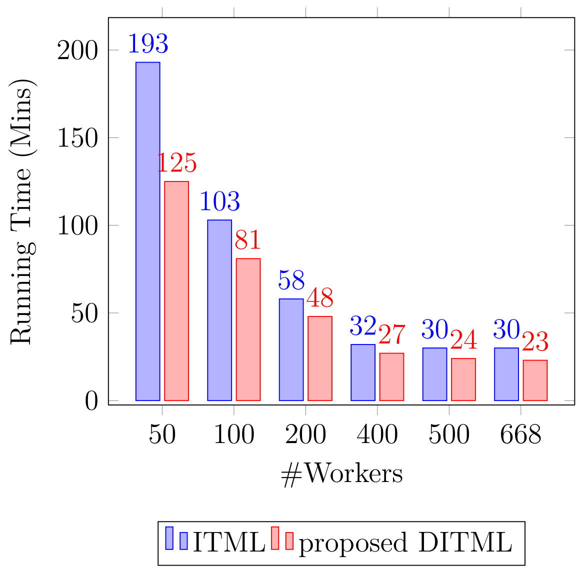


Figure 3.4: Comparisons between the original ITML and proposed DITML algorithms on 10000-dimensional synthetic data with different number of workers. Both of two algorithms iterates 50 times over all constraints.

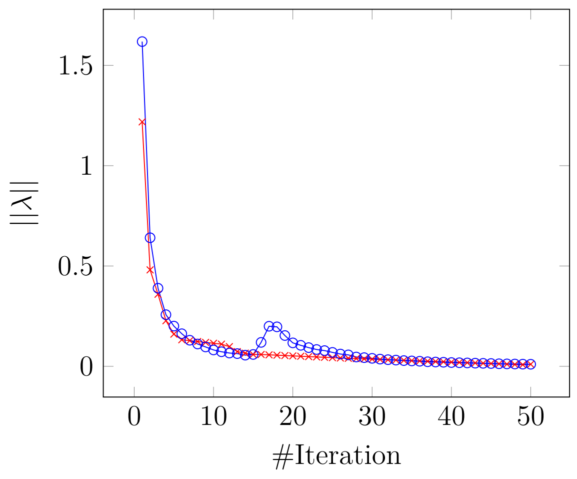


Figure 3.5: Convergence of the proposed DITML method and ITML on the ImageNet dataset. The iteration number here means the times that algorithms iterate over all the constraints.

classification and the accuracy gap between these two algorithms is relatively small.

3.5 Summary

In this chapter, based on the observation that the positive semi-definite Mahalanobis matrix can be decomposed into a series of rank-one matrices, we have developed a parallel distance learning algorithm called DITML in order to learn the proper distance function from high-dimensional data without low-rank approximation, which is infeasible for previous DML algorithms. Furthermore, the performance gap between the proposed method and the original ITML method is bounded from the theoretical perspective and can be ignored in actual experiments.

□ **End of chapter.**

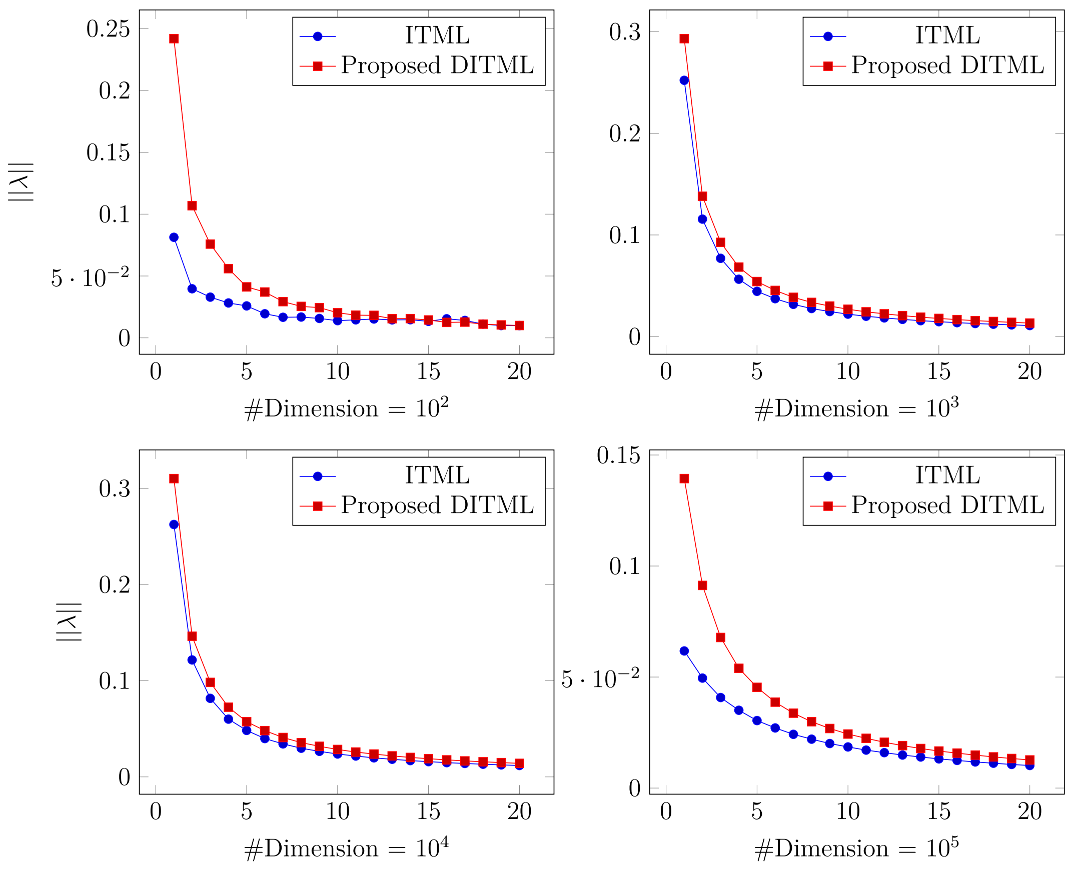


Figure 3.6: Convergence of ITML and the proposed DITML algorithms with different scaled synthetic dataset. The iteration number here means the times that algorithms iterate over all the constraints.

ALGORITHM 1: Parallel Distance Learning

```
Input: $S$ : set of similar pairs; $D$: set of dissimilar pairs; $u, l$ :
       distance thresholds; $\gamma$ : slack parameter
Output: $A$ : Mahalanobis matrix
$A = I$, $C = |S| + |D|$
for constraint $(x_p, x_q)_k, \quad k \in \{1, 2, \dots, C\}$ do
    $\lambda_k \leftarrow 0$
    $d_k \leftarrow u$ for $(x_p, x_q)_k \in S$;
    otherwise $d_k \leftarrow l$
    $c_k \leftarrow (x_p - x_q)_k$, $z_k \leftarrow c_k$
end
while $\beta$ does not converge do
    forall worker $k \in \{1, 2, \dots, C\}$ do
        $z_k = c_k + \sum_{i=1}^{C} \beta_i z_i z_i^T c_k$
        $p \leftarrow c_k^T z_k$
        if $(x_p, x_q)_k \in S$ then
            $\alpha \leftarrow \min\left(\lambda_k, \frac{1}{2}\left(\frac{1}{p} - \frac{\gamma}{d_k}\right)\right)$
            $\beta \leftarrow \frac{\alpha}{1-\alpha p}$
            $d_k \leftarrow \frac{\gamma d_k}{\gamma + \alpha d_k}$
        else
            $\alpha \leftarrow \min\left(\lambda_k, \frac{1}{2}\left(\frac{\gamma}{d_k} - \frac{1}{p}\right)\right)$
            $\beta \leftarrow \frac{-\alpha}{1+\alpha p}$
            $d_k \leftarrow \frac{\gamma d_k}{\gamma - \alpha d_k}$
        end
        $\lambda_k \leftarrow \lambda_k - \alpha$
        $z_k \leftarrow \left(I + \sum_{i=1}^{C} \beta_i z_i z_i^T\right) c_k$
        send $z_k$ to other workers.
    end
end
$A = I + \sum_{i=1}^{C} \beta_i z_i z_i^T$
```

	k-NN	ITML + k-NN	Proposed DITML + k-NN
Synthetic-10^2	0.900	0.930	0.920
Synthetic-10^3	0.940	0.962	0.957
Synthetic-10^4	0.933	0.938	0.938
Synthetic-10^5	0.812	0.923	0.900
ImageNet	0.682	0.835	0.830

Table 3.1: Accuracy of k-NN classification when $k = 4$

Chapter 4

Local Geometric Distance Learning for Information Retrieval

Conventional learning-to-rank (LtR) algorithms focus on query independent model, in which query and document do not lie in the same feature space, and the rankers rely on the feature ensemble about query-document pair instead of the similarity between query instance and documents. However, existing algorithms do not consider local structures in query-document feature space, and are fragile to irrelevant noise features. In this chapter, we propose a novel Riemannian metric learning algorithm to capture the local structures and develop a robust LtR algorithm. First, we design a concept called *ideal candidate document* to introduce distance learning algorithm to query-independent model. Previous distance learning algorithms aiming to find an optimal metric space are only suitable for query-dependent model, in which query instance and documents belong to the same feature space and the similarity is directly computed from the metric space. Then we extend the new and extremely fast global Geometric Mean Metric Learning

(GMML) algorithm to develop a localized GMML, namely L-GMML. Based on the combination of local learned distance functions, we employ the popular Normalized Discounted Cumulative Gain (NDCG) scorer and Weighted Approximate Rank Pairwise (WARP) loss to optimize the *ideal candidate document* for each query candidate set. Finally, we can quickly evaluate all candidates via the similarity between the *ideal candidate document* and other candidates. By leveraging the ability of distance learning algorithms to describe the complex structural information, our approach gives us a principled and efficient way to perform LtR tasks. The experiments on real-world datasets demonstrate that our proposed L-GMML algorithm outperforms the state-of-the-art distance learning to rank methods and the stylish query-independent LtR algorithms regarding accuracy and computational efficiency.

4.1 Problem and Motivation

In many information retrieval systems, especially Web search, users expect to obtain the most relevant documents according to users' query phrase or document. This task is technically formulated as a ranking problem. Most of the Web search engines exploit this ranking task based on learning-to-rank (LtR) techniques [79]. In the LtR framework, a machine learning algorithm is typically employed to derive a ranking model about document collection from a training subset of documents with labels or partial order. After the supervised or semi-supervised learning procedures, the ranking model is expected to retrieval top-k (ordered) relevant documents from the candidate collection when a query is given.

In practice, search engines develop the LtR model in two stages: (i) candidate retrieval and (ii) candidate re-ranking [80]. In the first stage, search engine retrieves from the inverted document repository a sufficiently large set of relevant candidate documents $\mathcal{D}_q$ matching a user's query. It is used to avoid applying the ranking model to all documents possibly matching a user's query. This stage usually requires that the size of candidate set is much larger than the number of the relevant URLs to be included in the returned page. Based on the candidate document set $\mathcal{D}_q$ obtained in the first stage, Web search engines reformulate the documents with features extracted from the query-document pair and hide query features, then employ the LtR model without the dependency of query instance to score and re-rank the document collection $\mathcal{D}_q$. Finally, search engines return the top-k documents to the user.

In Web search engine, the time-budget of this two-stage framework is usually limited. Therefore, strongly motivated by the time budget consideration, the current two most efficient and the state-of-the-art methods are based on the additive ensemble of regression trees, namely Gradient-Boosted Regression Tree (GBRT) [40] and λ-MART [16]. These two kinds of methods are capable of meeting the time requirement with acceptable accuracy even when thousands of regression tree are evaluated for each document. However, one of the drawbacks of this line of methods is that when the input samples contain an enormous amount of non-informative features, many methods fail to identify the most relevant features. Therefore, researchers are still trying to devise techniques and strategies to find a better way of combining features extracted from query-document pairs through discriminative training to accelerate the training process for document ranking without losing in quality [41, 131]. Another perspective of the ranking problem is to seek the best similarity measurement and develop the corresponding efficient algorithm. These approach aims to optimize the accuracy in the first stage to find candidate documents or even directly return the top-k documents with an order. Concerning accuracy, the similarity-based models for a ranking problem can be classified into three categories from the formulation of the loss function: point-wise, pairwise and list-wise loss functions [15]. Practically, the pairwise loss function tends to be more efficient for training and have been widely adopted by large Web search engines [15]. The pairwise similarity motivates that how to apply the classical distance learning or similar learning methods to the ranking problem [21]. The distance learning algorithms aim to find a better distance function than Euclidean distance to measure the pairwise similarity. The advantage of such distance-learning-to-rank [84] framework has two folds: (1) the distance functions often preserves the nearest neighborhood information, which is the perfect structure to conduct ranking; (2) a proper distance function containing the structural information of the document collections in the document space is useful for reducing the over-fitting and improving the robustness to noise features [64]. Therefore, the distance-learning-to-rank methods [76, 77, 84] typically enjoy higher accuracy. Nevertheless, unfortunately, the disadvantage of distance-learning-to-rank also has two folds: (1) many distance learning algorithms [29, 124] are degraded by its extremely high computational expense; (2) the similarity measurement is not suitable for LtR because we can not estimate the similarity between query and documents with features extracted from other domain knowledge.

In this chapter, we focus on improving the ranking accuracy at the second stage in the search engine and attempt to provide a new query-independent model for LtR task. Different from the existing research on how to combine features extracted from other domains, we try to learn an optimal representation of these features via distance learning algorithm. To adopt query-dependent distance learning framework to a query-independent model, we propose a concept called *ideal candidate document*, which represents a perfect match for a given query. With the help of this concept, we can quickly evaluate all candidate documents and sort them by calculating the distance based on the optimal distance function between the *ideal candidate document* and other documents. Same with the query-dependent model, the shorter distance leads to a higher relevance to the query.

Since features from different domains generate local structure on the whole feature space, in order to preserve more local information and avoid over-fitting, we develop a novel local distance learning framework for ranking with high efficiency and accuracy. Our localized distance learning algorithm extends from the state-of-the-art global distance learning algorithm called Geometric Mean Metric Learning (GMML) [135], and we apply Weighted Approximate Rank Pairwise (WARP) loss to optimize the distance function around the ideal document from the combination of several anchor documents.

We summarize our main contributions as follows:

- To the best of our knowledge, we are the first to extend geometric mean metric learning algorithm to a local distance learning approach in order to capture the local structures for LtR problem.

- We propose a novel *ideal candidate document* concept to transform distance-learning-to-rank framework from query dependent model to query independent model, which brings wider applications for distance learning and also improves the accuracy of classical LtR task.
- We conduct extensive experiments to demonstrate that our method outperforms the state-of-the-art query-dependent distance-learning-to-rank algorithms and query-independent LtR methods both in the accuracy and the computational complexity.

4.2 Methodology

In the information retrieval setting, a search engine maintains a collection of candidate examples $\mathcal{D}$. Given a query q, the search engine returns the top ranked subset of documents $\left\{p \in \mathbb{R}^d\right\} \subset \mathcal{D}_q \subset \mathcal{D}$ from the collection with order, ranked by a specific ranking model $f_q(p)$.
According to the formulation of the loss function, the LtR methods are categories into three folds: (1) pointwise loss approach, (2) pairwise loss approach and (3) listwise loss approach.
For pointwise loss function, Li et al. [74] cast the ranking problem to a multi-class classification problem. The training process relies on enough labeled information, which is not always easy to satisfy. Pairwise loss approach such as RankNet [14], RankBoost [38] focus on the relative order, which is capable of being adapted to classification problem. In the listwise loss approach, a relevance label l related with the query for ground truth is usually bound to the document p. Cao et al. [18] first propose to find the optimal permutation to minimize the listwise

loss function. McFee [84] proposes a similar objective, but the different solution from the distance learning methods.

The majority of LtR methods follows listwise loss function. Currently, the most popular methods [16, 30, 40] come from the combination of an ensemble of trees like random forest and the boosting-like methods [38]. Based on multiple decision trees, this kind of methods gains an accepted level of accuracy.

There are two different roadmaps to conduct the LtR tasks from the distance learning perspective: (1), McFee [84] and Lim et al. [76, 77] learn global distance function with Positive Semi-Definite (PSD) constraint on the metric parameter. They belong to the application of the standard distance learning algorithm. (2), Chechik et al. [21] and Liu et al. [78] remove PSD constraint and employ the bilinear model to measure the similarity between two data points. Usually, without PSD constraints, the bilinear model easily leads to over-fitting. However, PSD constraint brings a tremendous amount of computation.

In most cases, global distance learning relies on a learned PSD matrix, which is not only computational expensive in high-dimensional case but not reasonable for retrieval ranking problem. In the LtR framework, the local similarity is far more important than the dissimilar information because we aim at ranking the relevant documents around a user's query. Therefore, several important local distance learning approaches are related to our work. Wang et al. [118] parameterizes the weight function of each data point. The approach enhances the model complexity but brings extra computation. Hauberg et al. [51] provide the theoretical analysis about the optimal weight function. However, the calculation of the geodesics is extremely expensive.

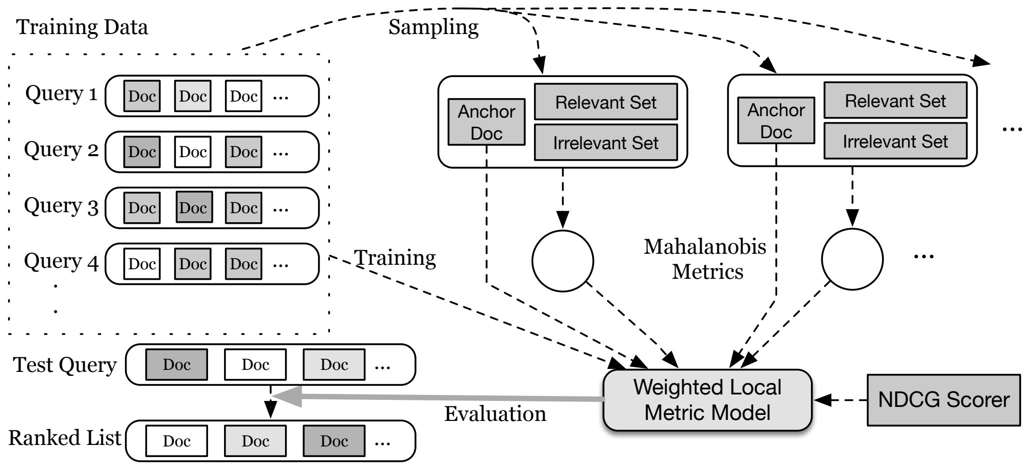


Figure 4.1: General framework of proposed L-GMML for ranking. Different gray levels in query test represent the relevant level of the document

In the LtR problem, a ranked list of the relevant documents is returned for a specific user's query. In this situation, we can assume without losing generality that all relevant documents should be closer to an unreal document than other irrelevant documents. This unreal document should be related to the query. Therefore, although the query instance is not accessible in the document feature space, we can still construct this unreal candidate document to represent the query in the feature space of the document. In our paper, this unreal but perfect-matching document is named as the *ideal candidate document.*

Usually, the indexed documents are assumed to be static, and the set of queries considered as input testing data is dynamic. This assumption allows us to transform the training documents to an another static representation, and model *ideal candidate documents* for each query to a dynamic combination of static documents.

In this thesis, we assume the documents including candidate documents and ideal documents lie on the surface of a Riemannian manifold. Then, we attempt to build the similarity measurement between documents on the geodesic lines in the Riemannian manifold. Very often, a single linear distance function $\mathbf{M}$ can not describe the whole surface of Riemannian manifold adequately. It reveals the inability of a single distance function to model the complexity of the LtR problem. Furthermore, the discriminative features vary between different neighborhoods on the surface of the manifold. To address this limitation, researchers try to learn a set of local distance function representing the various regions of the surface. In most cases, local distance learning algorithms will generate a local distance for each learning example [87]. The whole parameters of these kinds of the algorithm are prohibitively huge when the number of examples becomes large.

In our approach, we follow [118] to learn a local distance function for a part of the feature space of documents, in which case the number of learned distance function m can be considerably smaller than n, the size of the examples collection.

Suppose we have learned m local distance function $\{\mathbf{M}_1, \ldots, \mathbf{M}_m\}$ and the associated anchor points $\{p_1, \ldots p_r, \ldots p_m\}$. The choice of anchor points and the computation of local distance function are described in Subsection 4.2.1. Then the similarity model $f(q,p)$ between two documents p_i and p_j is extended from Eq. (2.9) as follows:

$$f(p_i, p_j) = d_{\mathbf{M}(p_i)}(p_i, p_j) \tag{4.1}$$

$$\mathbf{M}(p_i) = \sum_{r=1}^{m} w_r(p_i)\mathbf{M}_r, \tag{4.2}$$

where $w_r(p_i) \geq 0$ is the weight of document p_i for local distance function $\mathbf{M}_r$. The PSD constraints of $\mathbf{M}(p)$ is automatically satisfied if all local distance function $\mathbf{M}_r$ are PSD matrices. These formulation includes m anchor documents and p_i should be close to these anchor documents [94]. It is clear that the *ideal candidate document* should be close to several high relevant documents. Therefore, we can employ these high relevant documents as anchor documents to construct the local metric space around the *ideal candidate document*.

With the above assumption and observation, the task of information retrieval precedes in the following steps:

1. Given a candidate collection $\mathcal{D}_q$ for query q , we employ high/low relevant documents to compute a $\mathbf{M}$ and find a anchor point p_r to maximize the ranking scorer under the metric $\mathbf{M}$ by computing $(p_i - p_r)\,\mathbf{M}\,(p_i - p_r)^\top$.

2. After sampling m candidate collection to find m anchor documents and m associated distance functions, we can construct the *ideal candidate document* based on a combination of m anchor documents.

3. We can build an evaluation function to measure the similarity between candidate document and *ideal candidate document*, then, sort these documents via the similarity to *ideal candidate document*.

4.2.1 Computation of Basis Metrics

Before constructing the local distance function in Eq. (4.2), we need to learn m local metrics. With the assumption that each local metric $\mathbf{M}_r$ represents a part of feature space, we can employ the classical single distance learning algorithm associated with a subset of the triplets from a part of examples space.

In this chapter, we extend the state-of-the-art global distance learning algorithm GMML [135] into local distance learning forms. The extension consists of two parts:

1. The local basis metric associated with the triplets set $\mathcal{D}_r$ is computed by the original GMML.
2. The smooth weighting function $w_r(p)$ is computed from Eq. (4.10).

Given a subset of the triplets $\mathcal{T}_r = (p_i, p_j, p_k)$ such that p_i is more similar to p_j than to p_k, we can extract the similarity set S_r and the dissimilarity set D_r by following the instruction in Section 4.2.4. Then we construct two corresponding matrices:

$$\mathbf{S}_r = \sum_{(p_i,p_j)\in S_r} (p_i - p_j)(p_i - p_j)^\top \tag{4.3}$$

$$\mathbf{D}_r = \sum_{(p_i,p_k)\in D_r} (p_i - p_k)(p_i - p_k)^\top \tag{4.4}$$

Then, the basic optimization formulation of local metric $\mathbf{M}_r$ is defined as follows:

$$\min_{\mathbf{M}_r \succ 0} \quad h(\mathbf{M}_r) := \operatorname{tr}(\mathbf{M}_r\mathbf{S}_r) + \operatorname{tr}\left(\mathbf{M}_r^{-1}\mathbf{D}_r\right) \tag{4.5}$$

Equation (4.5) implies that GMML will return a single local metric $\mathbf{M}_r$ that minimize the sum of distances over all the similar pairs S_r and maximize the distance over all the dissimilar pairs D_r.

The closed-form solution of Eq. (4.5) is obtained by

$$\nabla h(\mathbf{M}_r) = \mathbf{S}_r - \mathbf{M}_r^{-1}\mathbf{D}_r\mathbf{M}_\mathbf{r}^{\mathbf{-1}} \tag{4.6}$$

Taking $\nabla h(\mathbf{M}_r) = 0$, we obtain:

$$\mathbf{M}_r \mathbf{S}_r \mathbf{M}_r = \mathbf{D}_r \tag{4.7}$$

Equation (4.7) is a Riccati equation whose unique solution is [135]

$$\mathbf{M}_r = \mathbf{S}_r^{-1/2} \left(\mathbf{S}_r^{1/2} \mathbf{D}_r \mathbf{S}_r^{1/2} \right)^{1/2} \mathbf{S}_r^{-1/2} \tag{4.8}$$

In experiments, $\mathbf{M}_r$ is efficiently computed from Cholesky-Schur method [57].

4.2.2 Smoothing Weight Functions

Lots of researchers try to provide the insights of their local distance learning approaches [51, 118] by modeling their methods from the perspective of Riemannian metric. An important property about the Riemannian metric is that a Riemannian metric $M(p)$ on a manifold $\mathcal{M}$ is a smoothly varying inner product $\langle x_i, x_j \rangle_p = x_i^T \mathbf{M}(p) x_j$ in the tangent space $\mathcal{T}_p\mathcal{M}$ of each point $p \in \mathcal{M}$. From Lemma 1 in [51], when the weight function $w_r(p)$ is smooth with p, Eq. (4.2) will be a well-studied Riemannian metric. Therefore, any well-designed local distance learning methods should provide a smooth weight function.

Another important issue is that the weight function $w_r(p)$ should reflect the fitness of the local metric $\mathbf{M}_r$. Suppose $(p, p_r) \in S_r$, it indicates that $\mathbf{M}_r$ is the best local distance function to measure the similarity between p_r and other examples, which means that Eq. (4.7) should be robust against the additive similar pair (p, p_r). Therefore, the weight function $w_r(p)$ should be in the opposite to $\mathbf{M}_r(p - p_r)(p - p_r)^T \mathbf{M}_r$.

Take the limit as an example, if $\mathbf{M}_r(p - p_r)(p - p_r)^T \mathbf{M}_r = \mathbf{0}$, then,

$$\mathbf{M}_r \left(\mathbf{S}_r + (p - p_r)\right) \left(\mathbf{S}_r + (p - p_r)\right)^T \mathbf{M}_r = \mathbf{D}_r \tag{4.9}$$

The solution of Eq. (4.9) is the same with Eq. (4.8), which indicates that $\mathbf{M}_r(p - p_r)(p - p_r)^T\mathbf{M}_r$ is a proper measurement whether the $\mathbf{M}_r$ is the optimal local distance function for the document p.

By taking consideration about the above observation, we propose the smoothing weight functions [12] as:

$$w_r(p) = \exp\left(-\frac{\rho}{2}\left\|p - p_r\right\|_{\mathbf{M}_r}\right), \tag{4.10}$$

where, $\|\cdot\|_{\mathbf{M}_r}^2$ is the L2 norm with the metric $\mathbf{M}_r$.

$$\|p - p_r\|_{\mathbf{M}_r}^2 = \operatorname{tr}\left(\mathbf{M}_r(p - p_r)(p - p_r)^T\mathbf{M}_r\right) \tag{4.11}$$

From Eq. (4.11) and Eq. (4.9), we can easily know $\|p - p_r\|_{\mathbf{M}_r}$ is a proper measurement about the similarity between query p and the anchor point p_r associated with the local metric $\mathbf{M}_r$.

Therefore, our evaluation function is formulated as:

$$f_q(p, \Phi_q) = -\sum_{r=1}^{m} \Phi_q^{(r)} \cdot \exp\left(-\left\|p - p_r\right\|_{M_r}\right) \cdot \left\|p - p_r\right\|_{M_r}, \tag{4.12}$$

where, $\Phi_q \in \mathbb{R}^m$, $\Phi_q^{(r)} = \exp\left(\rho_q^{(r)}/2\right)$ is the key parameter we need to learn in order to find a better manifold structure. Higher $f_q(p, \Phi_q)$ means that p is closer to the *ideal candidate document.* In the next subsection, we will introduce our exploration to optimize Φ for a specific ranking problem.

4.2.3 Update of Φ

In the above subsection, we formulate a general local distance learning framework in Eq. (4.12) to represent the manifold structure. From the theoretical analysis in [94], the whole space of Φ keeps the learned manifold smooth. Therefore, we define our loss function under the popular Weighted Approximate Rank Pairwise (WARP) framework [125] and optimize the associated objective function to obtain an optimal solution for ranking task.

The WARP loss for a given set of candidate document $\mathcal{D}_q$ with query ID q is defined as:

$$\mathcal{L}\left(q\right) = \frac{1}{\left|\mathcal{D}_q^+\right|} \sum_{p \in \mathcal{D}_q^+} L\left(v_q\left(p^+\right)\right), \tag{4.13}$$

where $v_q(p^+)$ is the number of violators in $\mathcal{D}_q$ for positive p^+, defined as:

$$v_q(p^+) = \sum_{p^- \in \mathcal{D}_q^-} \mathbf{I}\left[f_q\left(p^-, \Phi_q\right) - f_q\left(p^+, \Phi_q\right)\right] \tag{4.14}$$

To obtain better NDCG score, $L\left(\cdot\right)$ is defined as:

$$L(k) = \sum_{i=1}^{k} \frac{1}{\log_2\left(i+1\right)} \tag{4.15}$$

In order to optimize Φ_q, we follows the methods in [125, 76] to approximate $L\left(v_q(p^+)\right)$ by a continuous formulation with hinge loss:

$$\sum_{p^- \in \mathcal{V}_{q,p^+}} L\left(\left|\mathcal{V}_{q,p^+}\right|\right) \frac{\left[\zeta - f_q\left(p^+, \Phi_q\right) + f_q\left(p^-, \Phi_q\right)\right]_+}{\left|\mathcal{V}_{q,p^+}\right|}, \tag{4.16}$$

where for a given q, p^+, ζ is the hinge loss margin. $\mathcal{V}_{q,p^+}$ is the set of violators with hinge loss:

$$\mathcal{V}_{q,p^+} = \left\{p^- \in \mathcal{X}_q^- \mid f_q(p^+, \Phi_q)\right\} \tag{4.17}$$

In order to obtain an unbiased estimation of the loss function in Eq. (4.16), we can randomly sample q, $p^+ \in \mathcal{D}_q$ and find an violator p^- such that $\zeta + f_q\left(p^-, \Phi_q\right) > f_q\left(p^+, \Phi_q\right)$. In this situation, the tuple of (q, p^+, p^-) has the following contribution to Eq. (4.16):

$$l\left(q, p^+, p^-\right) = L\left(\left|\mathcal{V}_{q,p^+}\right|\right)\left(\zeta - f_q\left(p^+, \Phi_q\right) + f_q\left(p^-, \Phi_q\right)\right) \quad (4.18)$$

From the WARP framework, $\left|\mathcal{V}_{q,p^+}\right|$ can be approximated by $\left\lfloor\left|\mathcal{D}_q^-\right| / N_q\right\rfloor$, where N_q is the number of less relevant documents p^- drawn with replacement from $\mathcal{D}_q^-$ until a violator is found. Finally, the stochastic gradient descent for the parameter Φ can be easily conducted at iteration t as:

$$\begin{aligned} &\Phi_q(t+1) \\ =&\Phi_q(t) - \mu \frac{\partial l\left(q, p^+, p^-\right)}{\partial \Phi_q(t)}, && (4.19) \\ =&\Phi_q(t) - \mu L\left(\left\lfloor \frac{\left|\mathcal{D}_q^-\right|}{N_q} \right\rfloor\right) \cdot \left[\frac{\partial f_q(p^-, \Phi_q(t))}{\partial \Phi_q(t)} - \frac{\partial f_q\left(p^+, \Phi_q(t)\right)}{\partial \Phi_q(t)}\right], && (4.20) \end{aligned}$$

where $\frac{\partial f_q(p, \Phi_q)}{\partial \Phi_q} = \left[\frac{\partial f_q\left(p, \Phi_q^{(r)}\right)}{\partial \Phi_q^{(r)}}\right]_{r=1\ldots m}$. To avoid over-fitting, we project $\Phi_q^{(r)}$ to zero when Eq. (4.20) leads to negative value. We take derivation from Eq. (4.12) to obtain:

$$\frac{\partial f_q\left(p, \Phi_q^{(r)}\right)}{\partial \Phi_q^{(r)}} = \exp\left(-\left\|p - p_r\right\|_{M_r}\right) \cdot \left\|p - p_r\right\|_{M_r} \quad (4.21)$$

Overall, our proposed algorithm is illustrated in Figure 4.1 and summarized in Algorithm 3.

4.2.4 Sampling Strategy

Our approach will not iterate all triplets for $\mathcal{D}$ introduced in Section 4.2.1, because learning the global ranking model from all triplets is an NP-hard problem [83]. Hence, we choose to stochastically sample the triplets (p_i, p_j, p_k) from candidate documents. p_i and p_j representing similar documents are sampled from high relevant documents, then p_k is sampled from the less relevant documents. In our implementation, we only sample p_k from the documents with zero relevant label.
For sampling procedure in Section 4.2.3, Φ_i and Φ_j are independent for two queries i and j. Therefore, the update can be computed in a highly parallel way.

```
ALGORITHM 2: Geometric Mean Metric Learning (GMML) [135]
Input: $\mathcal{D}^+$ : positive set of documents, $\mathcal{D}^-$ : negative set of
        documents, $\lambda$ : regularization parameter
Output: $M \in \mathbb{S}_+^d$ : Mahalanobis metric;
$\mathbf{S} = \lambda\mathbf{I} + \sum_{p_i \neq p_j, p_i \in \mathcal{D}^+, p_j \in \mathcal{D}^+} (p_i - p_j)(p_i - p_j)^\top$;
$\mathbf{D} = \lambda\mathbf{I} + \sum_{p_i \in \mathcal{D}^+, p_j \in \mathcal{D}^-} (p_i - p_j)(p_i - p_j)^\top$;
$\mathbf{M} = \mathbf{S}^{-1/2}\left(\mathbf{S}^{1/2}\mathbf{D}\mathbf{S}^{1/2}\right)^{1/2}\mathbf{S}^{-1/2}$
```

4.3 Evaluations

In this section, we discuss the implementation of our approach for the LtR problem and display extensive experiments evaluating our methodology in comparison to the state-of-the-art (R-MLR, GBRT, and λ-MART). Our design on experiments tackle the following questions:

- Do we develop a correct localized extension to the global GMML? To answer this question, we generate varied scale

ALGORITHM 3: L-GMML to Rank

```
Input: Candidate set for $c$ queries $\{\mathcal{D}_1, \mathcal{D}_2, \dots, \mathcal{D}_q, \dots, \mathcal{D}_c\}$, $m$ :
       number of local metrics, $T$ : number of iteration, $\mu$ : step size,
       $\zeta$: hinge loss margin
Output: $\{(p_1, M_1), (p_2, M_2) \dots, (p_m, M_m)\}$ : set of local metrics and
        associated anchor points, $p \in \mathbb{R}^d$, $M \in \mathbb{S}_+^d$, $\Phi \in \mathbb{R}^{c \times m}$ :
        weights for local metrics to model the *ideal candidate
        documents* for each queries
for $q \in [1, c]$ do
    Extract $\mathcal{D}_q^+$ and $\mathcal{D}_q^-$ from $\mathcal{D}_q$;
end
for $i \in [1, m]$ do
    Sample $\mathcal{D}_i^+$ and $\mathcal{D}_i^-$ from $\{\mathcal{D}_q\}_{q \in [1,c]}$;
    $M_i = \text{GMML}\left(\mathcal{D}_i^+, \mathcal{D}_i^-\right)$;
    for $p \in \mathcal{D}_i^+$ do
        $\Gamma_p^{(i)} \leftarrow$Sort $\mathcal{D}_i$ in ascending order by computing
         $\|p - d\|_{M_i}^2 \ \ \forall d \in \mathcal{D}_q$;
    end
    Find the anchor point $p_r$ with maximum NDCG score of $\Gamma_{p_r}^{(i)}$;
end
for $t = 1$ to $T$ do
    Sample a tuple $(q, p^+, p^-)$ from $\{\mathcal{D}_q\}_{q \in [1,c]}$ such that
    $\zeta + f_q(p^+, \Phi_q(t)) > f_q(p^-, \Phi_q(t))$;
    $N_q \leftarrow$ the number of less relevant documents drawn with
     replacement from $\mathcal{D}_q^-$ until $p^-$ is found;
    $\Phi_q(t+1) = \left[\Phi_q(t) - \mu L\left(\left\lfloor \frac{|\mathcal{D}_q^-|}{N_q} \right\rfloor\right) \cdot \left[\frac{\partial f_q(p^-, \Phi_q(t))}{\partial \Phi_q(t)} - \frac{\partial f_q\left(p^+, \Phi_q(t)\right)}{\partial \Phi_q(t)}\right]\right]_+$;
end
```

synthetic datasets to evaluate the performance gain against global distance learning algorithm when a different number of local distance functions invoke in our L-GMML approach to prove the correctness.

- Is our assumption on the existence of local structures reasonable? If reasonable, does our solution enjoy high computational efficiency and the good scalability for scaled datasets? We make comparisons with the state-of-the-art distance learning algorithms for ranking in the query-dependent model on scaled datasets. We attempt to demonstrate the improvements of our approach over other distance learning algorithms.

- Does our LtR algorithm have any amazing properties? We conduct experiments on real-world large-scale datasets to illustrate the enormous improvement of our approach on accuracy compared with the dominant ranking methods like GBRT and λ-MART in the query-independent framework.

4.3.1 Experiments Setting

In our experiments, we have implemented our local GMML (L-GMML) algorithm in Julia [1], the source code is released at Github [2]. To make a fair comparison against the state-of-the-art ranking methods, we also implement R-MLR, GBRT and λ-MART in Julia. We take RankLib[3], an open-source implementation of the GBRT and λ-MART algorithms and MLR[4] as references to implement these algorithms in Julia.

[1]http://julialang.org/

[2]https://github.com/yxsu/LtR.jl

[3]http://sourceforge.net/p/lemur/wiki/RankLib/

[4]https://github.com/bmcfee/mlr

Our program is executed on an Ubuntu 14.04 LTS server with 12 Intel Xeon E5-2620 cores and 128GB main memory. All baseline methods and our method are performed in a parallel way to fully utilize the computational resources. Our R-MLR implementation is based on the parallel MLR-ADMM [77]. GBRT and λ-MART come from RankLib.
The statistical tests in the following experiments are computed over the values for Mean Average Precision (MAP) and Normalized Discounted Cumulative Gain (NDCG) [59] at the top k retrieved documents denoted as NDCG@k. These two metrics are the most important and frequently used in information retrieval community to evaluate a given permutation of a ranked list using binary and multi-relevance order.

4.3.2 Datasets

For all real-world datasets, we split each of them into two components: 1), the training set is used to learn ranking models; 2), the test set is purely used to evaluate the performance of the learned ranking models.
All the datasets we use are freely available online for scientific purpose. Such datasets can be divided into two groups:

Query-dependent Dataset

We employ CAL10K [112] to make fair comparisons between our L-GMML and R-MLR. Because, in the original paper, R-MLR performs well on the CAL10K dataset. Following the experiments in [77], we use a subset of the CAL10K dataset, which is provided as ten 40/30/30 splits of a collection of 5419 songs.

Table 4.1: Different kinds of song representation

	# of features	# of songs
Audio	1,024	5,419
Lyrics-128	128	2000
Lyrics-256	256	2000

Query-Independent Datasets

In this subsection, we employ two popular real-world large-scale datasets: Yahoo! and MSN to evaluate the competitive performance of proposed L-GMML against the state-of-the-art query-independent LtR methods.

Yahoo! datasets come from Yahoo! Learning to Rank Challenge [19]. The datasets consist of feature vectors extracted from query-url pairs along with relevance judgment labels.

In our experiments, we also employ the two set of MSN learning to rank[5] datasets: MSLR-10K and MSLS-30K, both of which consists of 136 features extracted from query-url pairs. The MSN datasets provide relevance judgment labels ranging from 0 (irrelevant) to 4 (perfect match). In experiments, each MSN dataset is partitioned into five subsets for five-fold cross validation.

The complete statistical information about these datasets are listed at Table 4.2.

4.3.3 Evaluation of the Proposed Approach

In our L-GMML model, the most important hyper-parameter is the number of local distance functions, which has significant influence on the overall model performance. We will evaluate

[5]https://www.microsoft.com/en-us/research/project/mslr/

the correction of our localized extension method from synthetic datasets, and reveal the impact of the metric numbers.

Global GMML vs Local GMML

In this subsection, we attempt to employ multi-class classification problem to verify the correction of the local distance learning algorithm. Because multi-class synthetic datasets certainly contain local structures around the center of each class. If the accuracy gain is observed, we can also address the objective that local distance learning approach is designed to extend the global distance learning method's ability of modeling complex data manifold.
Specifically, we employ the normal distribution to generate synthetic datasets with multiple centers and 95% confidence interval. The datasets with {10,50,100} classes are denoted as Synthetic-10, Synthetic-50, Synthetic-100 respectively. In these synthetic datasets, we assign the index of class to the relevant label of the corresponding data point.
We report the performance gain of the proposed local GMML against the global GMML in Figure 4.2. We can easily find the fact that when the number of local distance function is approximate to the number of the real centers in Gaussian synthetic data distribution, the relative accuracy gain of local distance function is maximized. This observation meets the objective of local distance learning approach.

The Number of Local Distance Functions

In this subsection, we will evaluate the significance of the number of local distance functions, which is typically the most important parameter in the field of local distance learning.

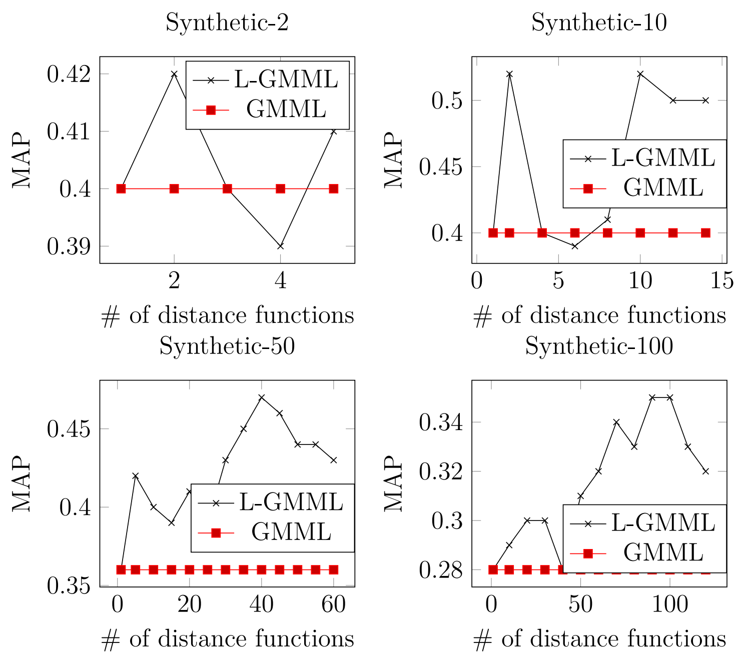


Figure 4.2: Comparisons between global GMML and local GMML on synthetic datasets. The performance is measured by MAP

A large number of local distance function will enhance the algorithm's ability to model the complex manifold structure. However the computational complexity increases linearly with the number of local distance function. In experiments, we need to carefully tune the number of local distance function to make the balance between model's ability and computational complexity.

Figure 4.3 displays the impact of the number of local distance function on all datasets used in our work. For all datasets, localized method can compete with the corresponding global method with a single distance function. This fact proves that our localized extension is reasonable. Another obvious observation is that the optimal number of local distance function varies dramatically among different datasets, since it is decided by the complexity of the manifold structure sealed in the data space.

Scalability

In our experiments, the synthetic datasets is primarily invoked to evaluate the scalability of our approach.
Due to the limited scalability of real-world datasets, we synthesize datasets with the feature dimensionality scaled from 10 to 1000. In this experiment, we fix the number of local distance function as 10 since we only concern about the computational complexity on different scaled dimensions instead of the optimal number of local distance function. Figure 4.4 illustrates the training time of our L-GMML on these datasets.
Compared with other local distance learning methods, the less training time come from two-fold issues: (1) the GMML in Algorithm 2 is very fast. (2) The update of weighting function in our approach is relatively simple and straightforward. It does not involve the huge computational resources to find the optimal form.

4.3.4 Comparison with R-MLR

The Robust Metric-Learning-to-Rank (R-MLR) [77] is the most competitive distance learning method for ranking. It retrieves relevant examples in response to a query instance. To make direct comparisons, we need to modify our approach by assigning all anchor points to the query instance. Because our approach is originally designed for the query-independent framework.
In this set of experiments, we evaluate our approach on the music similarity task, because the R-MLR method is verified to be successful in music similarity task compared with other metric-

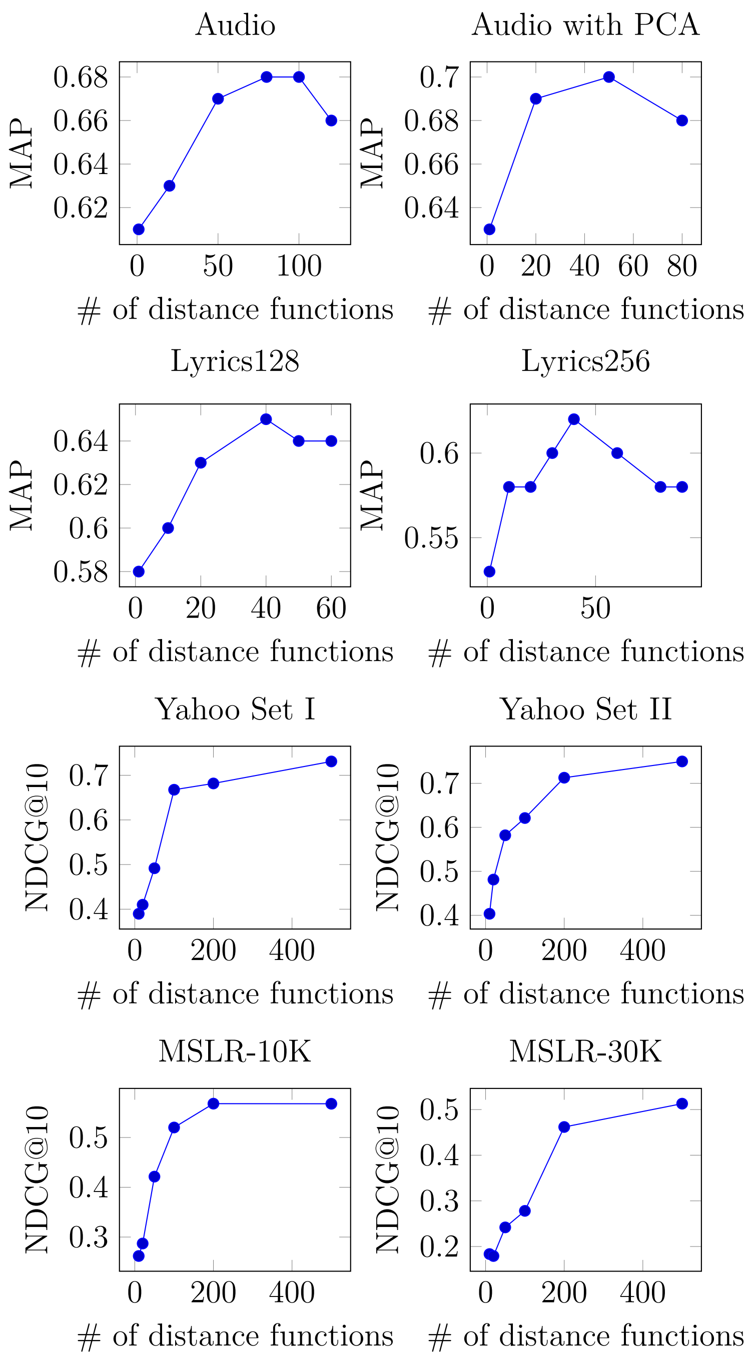


Figure 4.3: The variation of performance caused by different number of local distance function

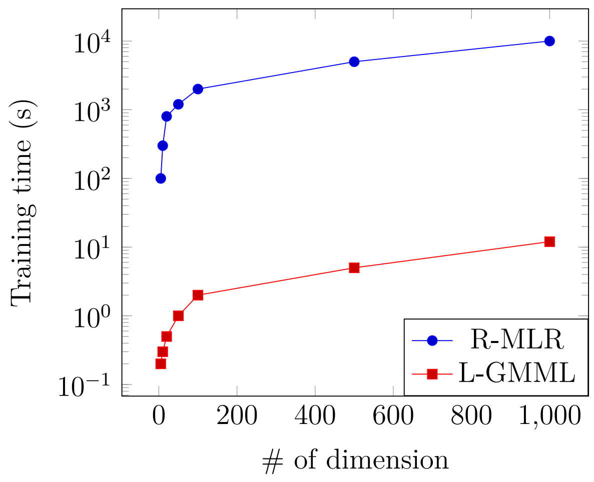


Figure 4.4: Training time of L-GMML on different scaled synthetic datasets

learning-to-rank methods such as MLR [84], $L1$-MLR [96]. For each song p_i, a relevant set $\mathcal{D}_i^+ \subset \mathcal{D}_{\text{train}}$ is defined as the subset of songs in the training set performed by the top 10 most similar artists to the performer of p_i, where the similarity between artists is measured by the number of shared users in a sample of collaborative filter data [84]. This top-10 thresholding results in the relevant sets in this data being asymmetric and non-transitive. Therefore, the traditional pairwise distance learning methods do not work in this situation. However, our approach based on the sampling on the relevant set is not necessary to obey the symmetric and transitive properties.

The experiments are conducted on two different kinds of song representation: audio and lyrics, whose details are listed in Table 4.1. We use recommended candidate hyper-parameters in the original paper to tune R-MLR on validation set and select the best parameter to evaluate the performance of the model.

Since the scalability of the original R-MLR is limited, the experiments of R-MLR employ the latent features compressed by PCA. Our approach has no such problem and is suitable to

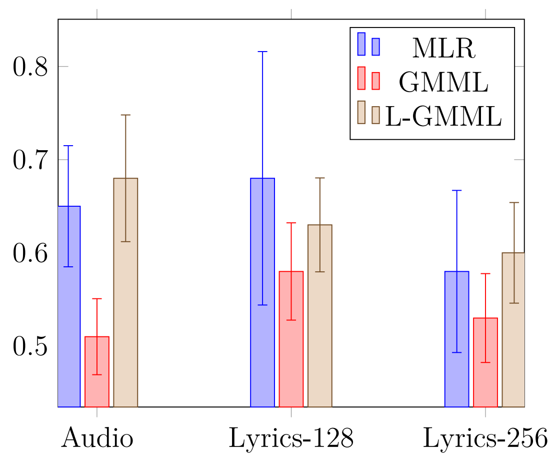


Figure 4.5: Music similarity performance of each algorithm on the three feature representation Audio, Lyrics-128 and Lyrics-256. Performance was measured by MAP and averaged across 10 folds.

conduct the training process on the original 1,024 features.
Figure 4.5 illustrates the performance of three distance learning algorithms. We fix the number of local distance function in our L-GMML as 1 to obtain the global GMML algorithm. The motivation of making such comparison is that we attempt to demonstrate the different influence of the new GMML algorithm and our proposed L-GMML algorithm on the performance improvements.
Therefore, we can draw the conclusion from the experiments in this subsection that the proposed approach outperforms other distance learning algorithms for the ranking problem regarding accuracy and computational efficiency.

4.3.5 Comparisons on Large-scale Real-world Datasets

We attempt to find amazing features of our method in the comparisons with two state-of-the-art ranking methods, Gradient-Boosted Regression Trees (GBRT) and λ-MART on Yahoo!

Set I&II, MSLR-10K, and MSLR-30K. Because they have been proved to be the most effective in the Yahoo! learning to rank challenge and become the dominant methods in the LtR field.

For these four datasets, the feature domain varies dramatically. To avoid for challenging the floating point precision in complex mathematical computation, we preprocess these four datasets by normalizing each feature dimension with 2-norm. For the stochastic sampling procedure in Algorithm 3, to find the optimal model, we try different initial weight values $\Phi(1)$ ranging from 0.1 to 10, the hinge loss margin ζ ranging from 0.01 to 1. The training time of GBRT and λ-MART is sensitive to the number of trees in both of the models. The number of local distance function also determines the training time of L-GMML. When we plan to make comparisons on the accuracy and training time of three methods, we fix the number of trees of GBRT and λ-MART as 5000 and the number of local distance function as 500. The motivation of these choices is that the performance of these two methods become stable on the four datasets. The comprehensive comparisons on a different measurement of the above three methods are illustrated in Table 4.4. From the table, we can draw a conclusion that our approach enjoys a huge advantage in accuracy compared with the state-of-the-art ranking methods.

Currently, the only disadvantage of our approach lies in scoring time. Table 4.5 displays the comparisons about the time of scoring documents. Our algorithm heavily relies on the scoring for each document in different stages, which is less efficient than GBRT and λ-MART. On the other hand, our approach is simple in structure, and GMML in the first stage is also efficient. Therefore, our method still has an advantage in computationally efficiency. The time-consuming comparison in Table 4.4 can prove this statement.

4.4 Summary

In this chapter, we focus on improving the accuracy of LtR methods by utilizing the local structure of documents and degrading irrelevant features. We firstly developed a localized GMML algorithm for the query-independent ranking framework. Specifically, we proposed a concept called *ideal candidate document* to adopt distance learning for ranking algorithm from a query-dependent model to widely used query-independent model. In our approach, a well defined smooth weighting function is optimized by reducing the popular WARP loss, which is defined for the candidate document set of a given query. Then we can efficiently score document by calculating the distance between candidate documents and a nonexistent *ideal candidate document* from an optimized distance function. The experiments prove that our approach outperforms both of the state-of-the-art query-dependent algorithms and query-independent algorithms.

□ **End of chapter.**

Table 4.2: Characteristics of publicly available large-scale datasets for learning to rank

Name	# of Queries			# of Doc.			Rel. Levels	# of Features	Year
	Train	Vali.	Test	Train	Vali.	Test			
Yahoo! Set I	19,944	2,994	6,983	473,134	71,083	165,660	5	519	2010
Yahoo! Set II	1,266	1,266	3,798	34,815	34,881	103,174	5	596	2010
MSLR-WEB10K	6,000	2,000	2,000	723,412	235,259	235,259	5	136	2010
MSLR-WEB30K	31,531	6,306	6,306	3,771k	6,306	753k	5	136	2010

Table 4.3: Comparison on the training time of R-MLR and L-GMML. The number of local distance functions in L-GMML is fixed as 50.

Time (s)	R-MLR	L-GMML
Audio	N/A	38
Audio with PCA	607	4.7
Lyrics-128	302	2.6
Lyrics-256	1241	7.8

Table 4.4: Performance of GBRT, λ-MART and proposed L-GMML on large-scale real-world datasets. Results of MSLR-WEB10K and MSLR-WEB30K are averaged from the 5 folds in the datasets.

Dataset		GBRT		λ-MART		L-GMML	
		Test Set	Time (min.)	Test Set	Time (min.)	Test Set	Time (min.)
Yahoo! Set I	NDCG@5	0.6529	41.2	0.6567	46.5	**0.6698**	28.1
	NDCG@10	0.6824	43.3	**0.7060**	48.0	0.6715	28.9
	NDCG@20	0.6912	41.5	**0.7091**	46.9	0.6934	28.8
Yahoo! Set II	NDCG@5	0.6731	37.6	0.6791	43.1	**0.7096**	26.5
	NDCG@10	0.6817	36.8	0.7062	43.3	**0.7264**	26.6
	NDCG@20	0.6954	37.4	0.7087	43.8	**0.7219**	26.4
MSLR-WEB10K	NDCG@5	0.4019 ± **0.0083**	49.4 ± 5.2	0.4417 ± 0.0131	58.3 ± 2.8	**0.4771** ± 0.0951	19.7 ± 2.1
	NDCG@10	0.4342 ± 0.0219	48.3 ± 2.1	0.4513 ± **0.0196**	57.6 ± 3.8	**0.5390** ± 0.0812	19 ± 3.1
	NDCG@20	0.4512 ± 0.0279	48.8 ± 3.8	0.4634 ± **0.0257**	57.1 ± 5.2	**0.551** ± 0.0728	19 ± 2.8
MSLR-WEB30K	NDCG@5	0.409 ± 0.0312	167 ± 28.6	0.3812 ± **0.0297**	182 ± 19.8	**0.4837** ± 0.0715	71.7 ± 2.7
	NDCG@10	0.4146 ± 0.0327	177 ± 30.1	0.409 ± **0.0232**	183 ± 17.9	**0.4976** ± 0.0619	71.9 ± 3.9
	NDCG@20	0.421 ± 0.361	167 ± 27.3	0.4112 ± **0.0240**	181 ± 10.7	**0.5038** ± 0.0718	72.5 ± 5.3

Table 4.5: Per-document scoring time of GBRT, λ-MART and L-GMML on Yahoo! and MSLR datasets. The scoring time is united in ms

	GBRT	λ-MART	L-GMML
Yahoo! Set I	276	302	421
Yahoo! Set II	218	286	421
MSLR-WEB10K	73	92	158

Chapter 5

Distributed Deep Distance Learning Framework with Hybrid Communications

Deep distance learning is widely used in extreme classification and image retrieval because of its powerful ability to learn the semantic low-dimensional embedding of high-dimensional data. However, the heavy computational cost of mining valuable pair or triplet of training data and updating models frequently in existing deep distance learning approaches becomes a barrier to apply such methods to a large-scale real-world context in a distributed environment. Moreover, existing distributed deep learning framework is not designed for deep distance learning tasks, because it is difficult to implement a smart mining policy of valuable training data. In this chapter, we introduce a novel distributed framework to speed up the training process of the deep distance learning using multiple machines. Specifically, we first design a distributed sampling method to find the hard-negative samples from a broader scope of candidate samples compared to the single-machine solution. Then, we design a hybrid communication pattern and implement a decentralized data-parallel framework to reduce the communication workload while the quality of the trained deep distance models is preserved. In experiments, we show excellent performance gain compared to a full spectrum of state-of-the-art deep distance learning models on multiple datasets in terms of image clustering and image retrieval tasks.

5.1 Problem and Motivation

Distance learning attempts to learn an advanced distance metric space in which the mapped representation of the raw data preserves the short distance between similar data points and the long distance between the dissimilar data points [6, 128]. This well-learned representation plays important role in information retrieval [109] and recommender systems [56]. For the past few years, with the success of deep learning, deep distance learning has received much attention. Because the distance learning shares similar assumption and objective with deep learning [69]. However, at odds with the conventional deep learning, which are successfully used to learn category related concepts on a large number of labeled data, deep distance learning aims to learn a nonlinear embedding of the data using deep neural network on a semantic metric space, which preserves the general concept of distance in data space.

Compared to deep distance learning, the drawback of the traditional deep neural networks such as GoogleNet [111] and ResNet [52] comes from the following two-fold: (a), the computational complexity of training and inference in deep learning is linearly proportional to the number of classes. It becomes impractical for extreme or fine-grained classification. (b), the most of deep learning models tend to overfit easily on a small amount of data per class. Therefore, deep distance learning yields promising results on many applications such as face recognition [99], zero-shot classification [13] and image retrieval [5].

The general procedure of deep distance learning contains two steps. First, a modified Siamese network [11] is optimized with the minimum value of one of the following loss: contrastive loss [49], random triplet loss [124], triplet loss with semi-hard negative mining strategy [99], lifted structured embedding loss [106], N-pair distance metric loss [104] and facility location loss [105]. Then, the semantic representing space generated from the Siamese network can be employed by many algorithms with efficient nearest-neighbor inference using the labeled data [54].

However, the existing deep distance learning approaches encounter a huge computational challenging for a large-scale dataset. Typically, deep distance learning considers a pair or a triplet of images as a training sample. Hence, n images can generate $\mathcal{O}\left(n^2\right)$ or $\mathcal{O}\left(n^3\right)$ training samples, which are intractable to iterate for large-scale dataset. A potential solution to reduce the size of search space is to find hard samples for training [26, 134]. Unfortunately, the hard sample mining brings enormous computation demands because it involves the inference for all data, which constantly varies during the training. To reduce the inference complexity, [26] involves human to label hard negative images from the evaluation of the model in each epoch. [120] samples triplets randomly during the first ten training epochs, then trains the model with negative hard triplets in each minibatch after ten epochs. Although a compromise between the inference complexity and the convergence process during the training is reached, the inefficient problem still exists in deep

distance learning framework.
Usually, distributed computation is a prominent tool to provide enough computational power. In the traditional deep learning community, distributed computation among multiple machines is also widely involved to handle the substantial computation demands [47]. Therefore, a deep distance learning oriented distributed algorithm is crucial to bring deep distance learning into big data applications. Since the smart sampling is a special issue in deep distance learning. It is difficult to utilize the conventional distributed deep learning platforms to find valuable training data globally. However, to our best knowledge, there is no distributed algorithm designed for or platform optimized for deep distance learning task.
In this chapter, to conduct deep distance learning in distributed environment efficiently, we first construct a distributed sampling method to find the valuable triplet training data efficiently among multiple machines. We analyze the characteristics of several deep distance learning algorithms, then construct an empirical framework to distribute the computation task of deep distance learning into machines. Precisely, a typical Siamese network consists of two cascading components: Convolutional Neural Network (CNN) part and the last layer part. In many deep distance learning algorithms, the first CNN part is initialized with a pre-trained model from traditional classification task. Based on the observation that the CNN part with the massive number of parameters varies infrequently compared to the frequently updated last layer part with fewer parameters, we have designed different communication patterns: ring topology for CNN part, All-Reduce topology for the last layer part to synchronize these two components among machines.

In summary, the contributions of this chapter are listed as follows:

- To the best of our knowledge, we are the first to propose a distance learning oriented distributed framework to speed up the training of deep distance learning among multiple machines.
- We have verified the decentralized distributed computation with mixed communication topology is reasonable in the context of deep distance learning.
- We also demonstrate the proposed framework is appropriately coupled with several state-of-the-art deep distance learning algorithms on CUB200-2011, CARS196, and Stanford Online Products and achieves a remarkable improvement with four machines regarding accuracy and runtime speedup.

5.2 Methodology

For the training process with large-scale datasets, we need larger mini-batch, partitioned into multiple machines. However, the deep distance learning with large mini-batch involves more communication demands because of the special selection method for mini-batch. In order to obtain a reasonable speedup with satisfying the accuracy, we need a smart architecture and policy for distributed training.

A common architecture for deep neural network systems takes advantage of data-parallelism [1]: a set of machines train model replicas on partitions of the input data in parallel. The model replicas are kept synchronized by all-reduced operation in Message Passing Interface (MPI) or parameter server, which maintains a global partition of the trained model and working machines fetch updated model from the parameter server periodically. These two methods guarantee the strong consistency of models among machines, but saturate the available network bandwidth, which becomes the bottleneck to accelerate the training process. In details, parameter server deteriorates the utility of the number of machines, because several physical machines are equipped with server maintaining global parameters. The network around the server easily becomes a bottleneck because of heavy communication tasks around the servers. For all-reduced operation, there is no centralized parameter server, but the frequent broadcast operations consume lots of network resources. Meanwhile, the computation unit must wait for the synchronization of large models. Therefore, it is an open problem that how deep neural network systems balance the use of computation and network resources to achieve the fastest model convergence.

Compared to traditional deep neural network, deep distance learning contains two unique features: (1), the selection of the mini-batch with valuable training data containing the hard-positive and hard-negative samples is costly. (2), the major component of the whole model is initialized with a sophisticated pre-trained model. To take advantage of the two features, we propose a distributed framework to conduct distributed sampling described in subsection 5.2.1 to find the valuable training data with hard-negative data globally. Then we design a mixed communication topology to synchronize these components with the different level of consistency, which is explained in the subsection 5.2.2.

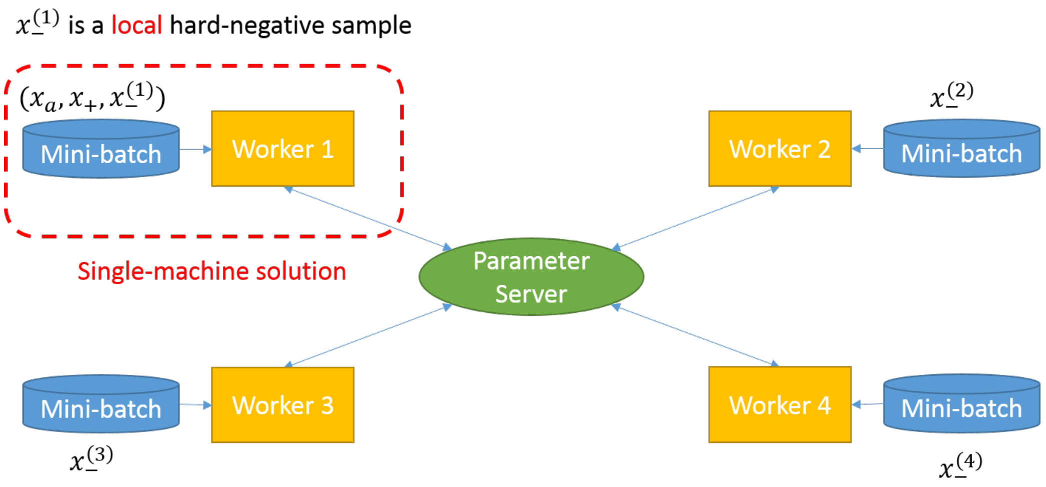


Figure 5.1: Illustration of the semi-hard negative mining in distributed environment.

5.2.1 Distributed Sampling and Evaluation

In this subsection, we first discuss the challenge of the adoption from single-machine solution to distributed solution for deep distance learning algorithm. Then, we proposed a distributed sampling and evaluation policy to alleviate the disadvantage of the single-machine solution.

Existing distributed deep learning frameworks only focus on the gradient synchronization (aggregation and scattering) among machines. There is no further consideration about data sampling methods, which are independent to machines in conventional platforms. Therefore, as illustrated in fig. 5.1, a straightforward conversion from single machine algorithm to distributed algorithm leads to a simple aggregation from several single-machine solutions. For each machine, they can not take advantage of a large portion of samples from other machines. This constrained sampling scope becomes a barrier to enhance the overall performance of deep distance learning algorithms.

Because a proper sampling strategy plays an equally important role as a suitable loss function [82].
In order to achieve a fast convergence rate, it is crucial to sample the hard positive and the hard negative samples from a large mini-batch. For a given anchor sample x_a, the hard positive is calculated from:

$$x_{+} \leftarrow \underset{i:y[i]=y[a]}{\arg\max} \left\| f\left(x_a\right) - f\left(x_i\right) \right\|_2^2 \tag{5.1}$$

The hard negative sample is calculated from:

$$x_{-} \leftarrow \underset{i:y[i]\neq y[a]}{\arg\min} \left\| f\left(x_i\right) - f\left(x_a\right) \right\|_2^2 \tag{5.2}$$

To mine the hard negative sample, we formalize our proposed method with optimization to identify a negative example from a large scope of mini-batch co-located in all machines. For machine p, suppose the number of classes in a mini-batch is $|C\left[p\right]|$. For each class, we randomly select an anchor sample x_a and then find the associated hard positive samples from the same class in the mini-batch. We denote the indicator set of the anchor sample for each class as $C\left[p\right]$. To support the above sampling rules, we should construct the mini-batch with more classes and keep the number of samples per class relatively constant.
The search for hard negative sample involves the online evaluation of deep neural networks, which is computationally demanding for a large number of instances from distinct classes. In our proposed framework, we dispatch the evaluation and comparison tasks to multiple machines. At the first stage, for each machine p, it will broadcast the embedding vector of anchor samples $\left\{f_p\left(x_a^{(i)}\right)\right\}_{i\in C[p]}$ to other machines. Note that the deep neural network model could be slightly different among machines, it

will be explained in the next subsection. We denote f_p as the deep neural network in the machine p.
At the second stage, each machine seeks for the hard negative sample corresponding to the incoming anchor samples. Specifically, for each machine q, it conducts the online evaluation of deep neural network for all samples in the mini-batch first, then calculates the distances between the embedding vectors and the incoming anchor samples to find the hard negative samples. For an anchor point x_{a_p} from the machine p, the hard negative sample is computed as:

$$x_-^{(a_p)} \leftarrow \arg\min_{i:y[i]\neq y[a_p]} \left\| f_q\left(x_i\right) - f_p\left(x_{a_p}\right)\right\|_2^2 . \tag{5.3}$$

When all hard-negative samples are found, the machine q will scatter the embedding vectors of these samples to other machines. Finally, when the set of triplet samples is ready in machine p, we can employ the back-propagation method with the following loss definition:

$$\begin{aligned} \ell\left([X]_p, [Y]_p\right) = -\frac{1}{\left|C\left[p\right]\right|}\cdot \\ \sum_{a\in C[p]} \log \frac{\exp\left\{f\left(x_a\right)^T f\left(x_{\mathcal{P}(a)}\right)\right\}}{\exp\left\{f\left(x_a\right)^T f\left(x_{\mathcal{P}(a)}\right)\right\} + \exp\left\{f\left(x_a\right)^T f\left(x_{\mathcal{N}(a)}\right)\right\}} \\ + \frac{\lambda}{m}\sum_i^m \left\|f\left(x_i\right)\right\|_2 , \end{aligned} \tag{5.4}$$

where $\mathcal{P}\left(\cdot\right)$ and $\mathcal{N}\left(\cdot\right)$ are the indicators of the hard positive sample and negative sample respectively. m is the size of mini-batch.

5.2.2 Hybrid Synchronization

Suppose there are P machines, machine i contains a cascading model parameterized by θ_i with two distinct components C_i and F_i, which denote the parameters of the CNN part initialized with the pre-trained model and the last fully-connected layer respectively.

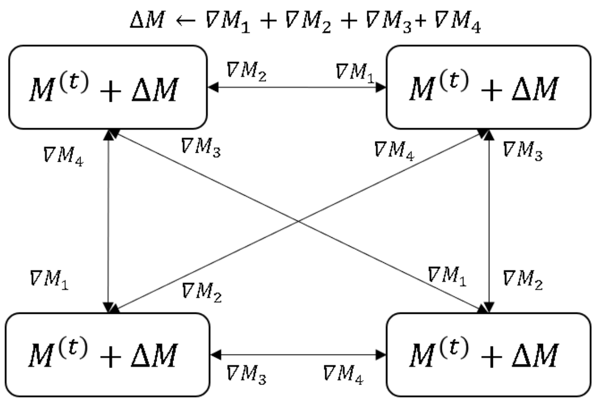
Typically, the parameters are calculated by the back propagation algorithm with gradient descent, which updates θ iteratively:

$$\theta_i^{(t+1)} \leftarrow \theta_i^{(t)} + \eta^{(t)} \sum_{j \in P} \nabla f \left(\theta_j^{(t)} \right), \tag{5.5}$$

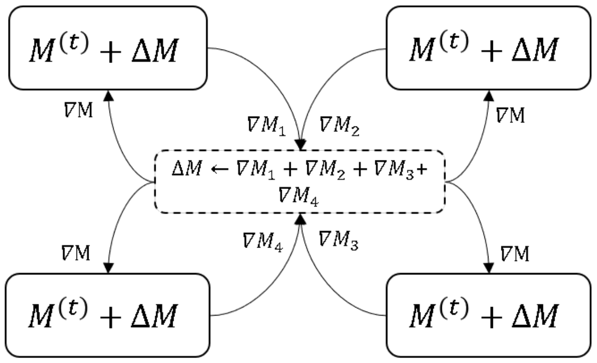
where θ_t are the parameters in iteration t, $\eta^{(t)}$ is the learning rate in iteration t.

In the setting of All-Reduce or the synchronous parameter server, $\theta_j = \theta_k \ \forall j, k \in [P]$. It requires the parameter server to collects all gradients and to conduct the gradient aggregation in Eq. (5.5), then broadcast the updated parameters to all machines. This strong consistency requirement brings a heavy burden for network communication and degrades the training speed accordingly.

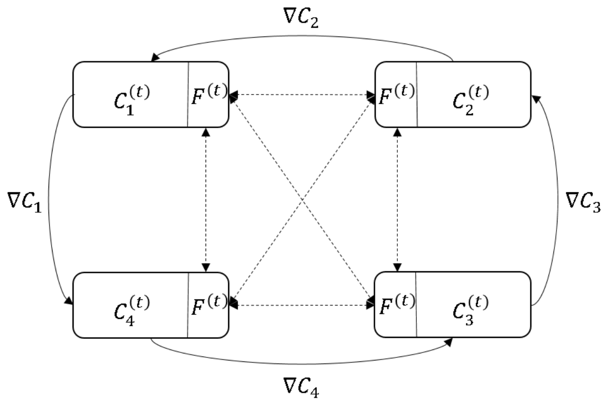
Empirically, we have the two following observations: (a), many deep distance learning algorithms are designed for two settings: end-to-end and last-layer. The latter means that the CNN part is initialized with a pre-trained model and then keeps fixed. (b), we observe that the average of ℓ_2 norm CNN part C_i varies insignificantly on many datasets, illustrated in fig. 5.5. Therefore, inspired by the decentralized framework [122], to increase the communication efficiency, we slacken the strong consistency constraints to allow the models in multiple machines are


$\Delta M \leftarrow \nabla M_1 + \nabla M_2 + \nabla M_3 + \nabla M_4$
$M^{(t)} + \Delta M$
∇M_1
∇M_2
∇M_3
∇M_4

(a) All-Reduce



(b) Parameter Server



(c) Hybrid Synchronization

Figure 5.2: Comparisons of several distributed communication topologies

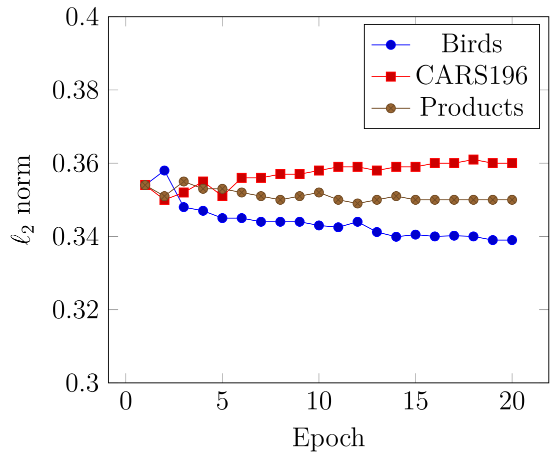


Figure 5.3: The average of the ℓ_2 norm of all parameters in CNN module during the training process.

different. We propose the ring-based synchronization topology as follows:

$$C_i^{(t+1)} \leftarrow C_i^{(t)} + \eta^{(t)} \left(\beta \nabla f \left(C_i^{(t)} \right) + (1-\beta) \nabla f \left(C_{\mathrm{Left}(i)}^{(t)} \right) \right), \quad (5.6)$$

where Left $(\cdot)$ denotes the index of last machine in a ring topology, $\eta^{(t)}$ represent the learning rate at iteration t. Currently, $\beta = 0.5$ is a predefined constant.

Ring-based synchronization is an optimal bandwidth solution to synchronize the parameters between machines, because the communication complexity of each machine reduces from $\mathcal{O}(P)$ to $\mathcal{O}(1)$. In experiments, we have profiled the training process to monitor the change of all CNN components $C_{[P]}$. Figure 5.5 demonstrates the convergence between any two adjacent machines in the ring topology.

For the last layer, we follow the strong consistency requirements to update the last layer F_i, because the number of parameters in F_i is relatively small and the synchronous update does not barrier the training process.

In overall, the proposed hybrid synchronization consists of ring-based synchronization to update the large amount parameters in CNN models, and All-Reduce synchronization to update the frequently changed parameters in the last layer. Figure 5.2 illustrates the difference of the three communication topologies. Algorithm 4 summaries the proposed distributed deep distance learning algorithm.

ALGORITHM 4: Proposed Distributed Deep Distance Learning Algorithm

```
Input: Image datasets with $C$ labeled categories, $P$ : number of
       machines, $T$ : number of iteration, $\eta$ : step size, $\lambda$:
       regularization parameter
Output: $f(\theta)$: optimized deep neural network parameterized by $\theta$
for $t = 1, \ldots, T$ do
    forall machine $p \in [1, P]$ do
        Randomly sample to from the whole dataset;
        Randomly select anchor sample $x_a$ for each class. $C[p]$
          represents the indicator set of anchor sample for each class;
        Find the hard positive sample by:
          $x_+ \leftarrow \arg\max_{i:y[i]=y[a]} \|f(x_a) - f(x_i)\|_2^2$
        Broadcast $\left\{f_p\left(x_a^{(i)}\right)\right\}_{i \in C[p]}$ to other machines;
        Receive $\left\{f_q\left(x_a^{(i)}\right) : i \in C[q], q \in [1, P]\right\}$ from other machines;
        Find all hard negative samples by:
          $x_-^{(a_q)} \leftarrow \arg\min_{i:y[i] \neq y[a_q]} \left\|f_p(x_i) - f_q\left(x_{a_q}\right)\right\|_2^2 \quad \forall a_q \in C[q], q \in [1, P]$;
        Scatter the above hard negative samples;
        After receiving the hard negative samples, conduct
          back-propagation on the loss defined in Eq. (5.4) to update $\theta$;
        All-reduce the FC parameters $F_p$ among all machines;
        Send CNN parameters $C_p$ to its right neighbor Right($p$);
        Update $C_p$ with Eq. (5.6);
    end
end
```

5.2.3 Implementation Details

We have implemented our framework [1] with the popular PyTorch [88]. In order to reduce the waiting time of gradient synchronization, we modified the default PyTorch architecture to allow us to synchronize the gradients of the current layer in an asynchronous way while computing the gradients of the next one simultaneously.

For network architecture, we basically follow the configuration in [105]. We employ existing network with pretrained parameters and finetune the network on new datasets. At the end of deep neural network, we conduct ℓ_2 normalization first to the embedding vector before computing the loss. The experimental ablation study in [106] reports that the dimensionality of the embedding vector does not play a crucial role during the training and testing process. We explored the dimensionality of embedding vector from 64 [105] to 512 [106] . The larger embedding size will perform slightly better than the smaller ones in accuracy but bring heavy communication requests in a distributed setting. Therefore, we select 128 as a trade-off. For each iteration, we only need to sample m examples and labels from different classes at random. In particular, we randomly select 5 images per class in a mini-batch. There is no need to prepare the data in any rigid paired or triplets format.

For the network architecture, the CNN components are initialized with the ResNet-34 [52] pre-trained on the popular ImageNet / ILSVRC 2012-CLS dataset. The final fully-connected layers FC are randomly initialized.

We fine-tuned the network on the training datasets with two configurations:

[1] `https://github.com/yxsu/ddml-hs`

- End-to-End: we fine-tune the overall model and update the parameters of all the layers during the back-propagation with the proposed hybrid synchronization. In this case, we perform 20 epochs of gradient descent.
- Last Layer: we keep all CNN components pretrained on ImageNet unchanged and update the last layer with our hybrid synchronization. In this situation, the communication in our framework is equivalent to the default All-reduce pattern in PyTorch.

All the input images are resized to 256×256 and cropped at 227×227. We use a random crop with random horizontal mirroring for training and a single center crop for testing. [104] takes multiple random crops and compute the average from the cropped images as the embedding vector.

5.3 Evaluations

In experiments, we attempt to answer the following two questions: (a), is our proposed distribution framework correct in the context of deep distance learning? We can obtain the answer from the comparisons between proposed framework and the several existing deep distance learning algorithms regarding accuracy in clustering and image retrieval. (b), is our framework communication-efficient and does it enjoy good scalability regarding the number of the machines? The observation on the execution time of the proposed framework with different hardware setting will illustrate the efficiency issue.
In the experiments, we evaluate our distributed framework on the following widely-used fine-grained datasets and employ the

same train/test splits.

- The Caltech-UCSD Birds (CUB200-2011) [117] consists of 11,788 images of birds from 200 different species/categories. We use the standard configuration in which the first 100 categories (5,864 images) is used for training and the rest for test (5,924).
- CARS196 [63] consists of 16,185 images of cars from 196 model categories. The first 98 categories (8,054 images) are used for training and the rest for test (8,131 images).
- Stanford Online Products [106] consists of 120,253 images from 22,634 online product categories. The default configuration contains 59,551 images from 11,318 categories for training and 60,502 images from 11,316 categories for test.

In these experiments, we choose to disjoint the categories for training and testing separately, although they belong to the same context (they all represent birds, cars or products). This makes the problem more challenging because traditional deep neural network models may easily overfit on the training datasets with a vast number of categories.

All experiments are conducted on 4 servers with 2 Pascal GPUs each connected by 10 Gbit/s network. To make fair comparisons, we closely follow [106, 105] for deep distance learning configuration in experiments.

For clustering evaluation, we calculate the embedding vector on all the test images first, then conduct affinity propagation clustering algorithm [39] with bisection method for the predefined number of clusters, which is equal to the number of classes in the test set. We use the standard Normalized Mutual Information

(NMI) metrics to quantitatively measure the clustering quality [105]. NMI is defined by the ratio of mutual information entropy of clusters and labels.

For the image retrieval evaluation, we employ the standard Recall@K metric in the standard K nearest neighbor retrieval task. Recall@K is defined as the fraction of queries for which there exists a data sample of the same class as that of the query instance with the first K positions of the retrieved list.

We compare our proposed methods against several state-of-the-art methods including triplet loss with semi-hard-negative mining [99], deep distance learning via lifted structure [106], deep distance learning with histogram loss [114], deep distance learning with N-pair loss [104], deep distance learning via facility location (also called NMI-based) [105] and deep spectral clustering [68]. In experiments, to make completed comparisons, we have also adopted the state-of-the-art methods from single-machine solution to distributed computation environment with All-reduce communication topology, which is the standard distributed implementation in TensorFlow [1] and PyTorch [88].

5.3.1 Convergence

In this subsection, we attempt to show the correctness of our proposed framework on all three datasets.

In our proposed framework, the deep neural network model could be slightly different with its neighbors. Since we loose the consistency requirement in our hybrid synchronization policy to the decentralized setting. The theoretical analysis on the convergence issue of this decentralized deep neural network models is still an open problem [75]. In this chapter, we only conduct an empirical study on this convergence issue. Figure 5.5

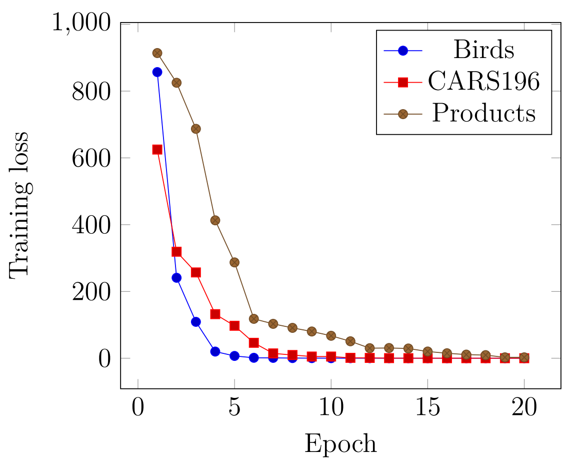


Figure 5.4: The training loss of the averaged deep neural network model in our proposed framework with 4 machines.

illustrates the difference of models among neighbors on three datasets. From the figure, we can see that the ℓ_2 norm difference of the parameters in CNN model convergence to zero after 10 epochs. It means that all parameters among neighborhood convergences to the same values.
Figure 5.4 further demonstrates the convergence process of our proposed framework. During the training process, we take the average of all training loss in different machines as the overall training loss depicted in fig. 5.4. After training, we consider the averaged model as the final optimized model to conduct test tasks.

5.3.2 Quantitative Results

In this subsection of quantitative evaluation, we try to empirically prove the effectiveness of our proposed distributed deep distance learning method in image retrieval and clustering tasks. Tables 5.1, 5.2 and 5.3 display the comparison results between several state-of-the-art methods and our proposed method

Table 5.1: Evaluations on the Caltech-UCSD Birds (CUB-200-2011) dataset. Recall@k is the recall with the top k results.

Method	Setting	Time (s) / epoch	NMI	Recall@1	Recall@4	Recall@8
Triplet semi-negative [99]	Original	524	56.12	40.46	58.15	69.28
	2 machines	458	56.10	40.98	57.45	69.56
	4 machines	362	56.98	41.05	58.34	69.17
Lifted struct [106]	Original	536	56.30	43.24	66.73	79.61
	2 machines	441	56.20	44.19	66.50	79.13
	4 machines	391	56.98	44.34	66.56	79.78
Histogram [114]	Original	520	-	49.34	68.61	80.58
	2 machines	437	-	49.10	66.97	78.24
	4 machines	358	-	48.34	65.10	77.10
N-pairs [104]	Original	513	58.87	44.29	67.26	79.18
	2 machines	420	58.97	45.14	67.26	79.78
	4 machines	340	59.20	45.19	67.98	80.10
NMI-based [105]	Original	702	60.19	48.38	72.47	82.25
	2 machines	599	60.15	48.19	72.38	81.98
	4 machines	493	60.17	48.56	72.50	82.01
Spectral [68]	Original	617	58.13	50.28	76.80	**85.79**
	2 machines	574	58.10	50.78	76.75	85.58
	4 machines	432	58.98	51.03	76.85	85.78
Ours	2 machines	378	61.19	52.45	77.08	84.24
	4 machines	**234**	**61.96**	**52.87**	**78.19**	85.07
	4 machines (last layer)	**86**	55.34	43.19	60.73	75.21

Table 5.2: Evaluations on the CARS196 dataset. R@k is the recall with the top k results.

Method	Setting	Time (s) / epoch	NMI	Recall@1	Recall@4	Recall@8
Triplet semi-negative [99]	Original	507	54.09	41.30	75.21	80.32
	2 machines	446	54.13	44.25	75.59	79.19
	4 machines	378	54.20	44.56	75.50	79.31
Lifted struct [106]	Original	530	56.90	53.70	75.98	83.30
	2 machines	439	57.02	53.98	76.04	83.34
	4 machines	395	57.13	53.88	76.05	83.50
Histogram [114]	Original	512	-	53.67	75.56	81.20
	2 machines	425	-	53.10	74.97	81.09
	4 machines	355	-	53.00	74.56	80.78
N-pairs [104]	Original	489	58.04	54.36	79.03	84.23
	2 machines	398	58.34	54.56	79.01	84.34
	4 machines	336	58.78	54.88	79.12	84.56
NMI-based [105]	Original	832	57.27	57.29	79.90	88.24
	2 machines	703	57.17	57.09	79.78	88.05
	4 machines	578	57.20	57.19	79.81	88.20
Spectral [68]	Original	798	61.08	69.35	81.08	90.35
	2 machines	637	61.10	69.56	81.10	90.29
	4 machines	501	61.21	69.70	81.23	**90.40**
Ours	2 machines	423	61.24	70.07	82.13	89.25
	4 machines	**298**	**61.80**	**70.73**	**82.87**	90.10
	4 machines (last layer)	**93**	53.10	44.37	77.19	82.28

Table 5.3: Evaluations on the Stanford Online Products dataset. R@k is the recall with the top k results.

Method	Setting	Time (s) / epoch	NMI	Recall@1	Recall@4	Recall@8
Triplet semi-negative [99]	Original	4,097	88.38	67.12	77.97	82.28
	2 machines	3,658	88.40	67.23	78.01	82.38
	4 machines	3,167	88.42	67.34	78.01	82.38
Lifted struct [106]	Original	3,814	88.19	65.50	78.23	81.08
	2 machines	3,512	88.22	65.68	78.19	81.00
	4 machines	3,170	88.25	65.71	78.41	81.56
Histogram [114]	Original	3,856	-	65.55	78.73	81.56
	2 machines	3,550	-	65.01	78.30	81.12
	4 machines	3,210	-	64.95	78.15	81.03
N-pairs [104]	Original	3,701	89.01	67.12	79.15	84.09
	2 machines	3,506	89.12	67.31	79.21	84.15
	4 machines	3,010	89.18	67.35	79.51	84.34
NMI-based [105]	Original	6,238	90.27	66.98	77.06	82.15
	2 machines	5,078	90.34	67.10	77.10	82.20
	4 machines	3,998	**90.35**	67.28	77.21	82.24
Spectral [68]	Original	5,713	87.38	66.09	78.98	83.12
	2 machines	4,892	87.50	66.12	79.01	83.20
	4 machines	3,820	87.55	66.15	79.23	83.20
Ours	2 machines	3,028	89.17	67.79	80.34	**84.73**
	4 machines	**2,356**	89.56	**68.02**	**80.48**	84.70
	4 machines (last layer)	**898**	86.57	64.14	76.23	80.18

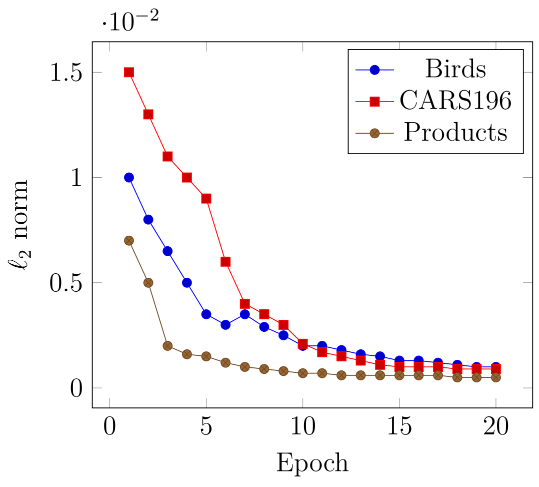


Figure 5.5: The average of $\left\|C_i - C_{\text{Right}(i)}\right\|_2 \forall i \in [P]$. This figure displays the convergence of CNN models with 4 machines

in terms of the NMI and k nearest neighbor task with the Recall@K metric. In this subsection, we only consider the "Original" setting for the state-of-the-art methods. "Original" means that all these algorithms are executed in the original single-machine solution with the same configuration in their original paper. However, we do not report the results of our proposed framework in the single machine setting. Because in a single machine, our sampling policy cannot take advantage of the large portion of data samples in a mini-batch. Therefore, the comparisons in the single-machine setting are unfair. The overall performance of our method in this environment is worse than the N-pairs method [104] with the similar loss definition. From the tables, we can easily see a considerable improvement of our proposed method compared to the state-of-the-art methods, especially the N-pairs method. The improvement against N-pairs method demonstrates the effectiveness of our distributed sampling policy. Because our method has a similar loss definition compared to the N-pairs method and the significant

difference comes from the extension of the sampling policy to the distributed configuration.
For the execution time, table 5.1, 5.2 and 5.3 also demonstrate distinguishable results from all these state-of-the-art methods. The execution time of the triplet loss with semi-hard-negative mining , the N-pairs method and the lifted structure method are comparable. Because these methods spend the majority of time on the online evaluation of the deep neural network. Compared to other methods, NMI-based method is slowest for each epoch. This is due to the cost of finding the facility location in this method.
For different datasets, the evaluation time of these methods for each mini-batch on the larger Stanford Online Products dataset is comparable to that on the smaller CUB-200-2011 and CARS196 datasets. Since all input images are scaled into the same size before evaluating the deep neural network. However, all methods are slow on the Stanford Online Products dataset. Because the Stanford Online Products dataset is larger than the others and has more rounds of mini-batching for each epoch. For the last-layer setting, as demonstrated in table 5.1, 5.2 and 5.3, the execution time is very fast. Since there is no computationally expensive back-propagation and heavy communication demands for the CNN components. However, in most cases, the performances of our proposed method with the last-layer setting on three datasets are not competitive concerning image retrieval and clustering tasks.

5.3.3 Scalability

In this subsection, we attempt to demonstrate the scalability of several state-of-the-art methods with the distributed setting.

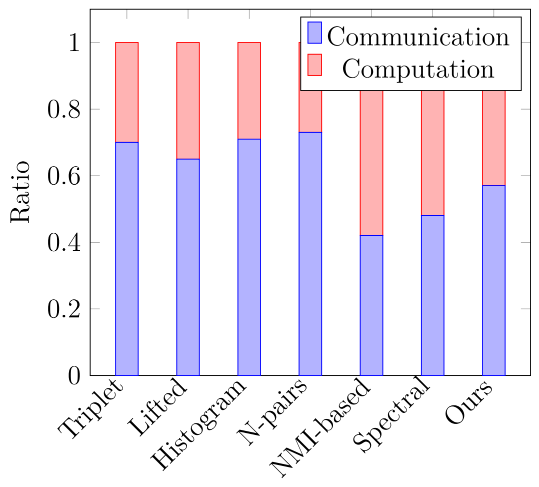


Figure 5.6: Comparison of computation time and communication time on the Caltech-UCSD Birds dataset with 4 machines. The results from other two datasets are similar.

Then, we compare our proposed framework and the other methods with different distributed settings, in which the different behaviors motivate our work.

In triplet framework, large batch size brings a large scope of candidate samples, and it enjoys high probability to generate useful triplets with real hard-positive and hard-negative samples. Furthermore, there is generalization gap between large batch size and small batch size in learning theory. However, since existing deep distance learning algorithm are all designed for single machine environment, which contains a limited number of GPUs. The limited GPU memory constraints the maximum batch size as well as the model complexity of deep neural network. For example, we believe the overall performance of the state-of-the-art methods will increase if we only replace ResNet-34 with ResNet-151 and keep sampling policy unchanged in these methods. Unfortunately, the higher complexity of deep neural network will cost more GPU memory and lead to a

smaller mini-batch size owing to the limited total GPU memory. Therefore, a suitable distributed framework is necessary for deep distance learning algorithm.
From table 5.1, 5.2 and 5.3, we can see that the overall execution time of all methods decreases when the number of machines increases. Specifically, the computation time for each epoch decreases. Because the rounds of mini-batching in each machine is less than that in single machine. However, the communication illustrated in fig 5.6 is costly. The strong consistency requirement in All-reduce distributed framework brings heavy communication workload in this case. For the performance in image retrieval and clustering tasks, we can not observe a stable improvement for the state-of-the-art methods when the number of machine increases. Therefore, a simple gradient aggregation policy implemented in All-reduce framework is not benefiting to distributed deep distance learning. This phenomenon is different from the conventional distributed deep learning approaches. The potential improvement should come from a smart sampling method which utilizes all machine resources.
Our proposed framework performs well in image retrieval and clustering tasks. In many cases, it achieves the state-of-the-art results. For the scalability issue, our proposed method performs better in general when the number of machine increases. These experiments prove that our distributed sampling method has a positive impact on the overall performance of deep distance learning algorithms.
For communication time, we expect that the ratio of communication time is as low as possible. Figure 5.6 illustrates the ratio of communication time and computation time of several methods. From the figure, we can easily observe that the ratio

of communication time in our proposed method is lower than many state-the-of-art methods in a conventional distributed framework. The advantage of our proposed framework is more obvious if the algorithm is conducted on a larger-scale cluster with more machines. For NMI-based and Spectral methods, the lower ratio of communication time does not reveal an excellent performance on a distributed framework. In fact, the computation of NMI-based and Spectral methods are more costly.

5.4 Summary

In this chapter, we present a scalable solution from the perspective of distributed computation to tackle the big data challenge for deep distance learning algorithm. In particular, we first propose a distributed sampling method to find the hard-negative samples from all machines to improve the efficiency of triplet mining method. Then, with the help of the unique characteristics of deep distance learning, we further propose a hybrid synchronization policy to speed up the training process of deep distance learning algorithm by reducing the communication workload significantly in a distributed environment. Finally, extensive experimental results demonstrate the effectiveness of our proposed distributed deep distance learning framework on image retrieval and clustering tasks.

□ **End of chapter.**

Chapter 6

Distributed Barycenter Estimation for Wasserstein Distance

Wasserstein Generative Adversarial Nets (GANs) are newly proposed GAN algorithms and widely used in computer vision, web mining, information retrieval, etc. However, the existing algorithms with approximated Wasserstein loss converge slowly due to heavy computation cost and usually generate unstable results as well. In this chapter, we solve the computation cost problem by speeding up the Wasserstein GANs from a well-designed communication efficient parallel architecture. Specifically, we develop a new problem formulation targeting the accurate evaluation of Wasserstein distance and propose an easily parallel optimization algorithm to train the Wasserstein GANs. Compared to traditional parallel architecture, our proposed framework is designed explicitly for the skew parameter updates between the generator network and discriminator network. Rigorous experiments reveal that our proposed framework achieves a significant improvement regarding convergence speed with comparable stability on generating images, compared to the state-of-the-art of Wasserstein GANs algorithms.

6.1 Problem and Motivation

Generative Adversarial Networks (GANs) [46] have recently achieved significant progress for numerous applications such as computer vision [126], information retrieval [119, 109] and recommender system [140, 139, 138]. GANs consist of two networks: a generator network that creates plausible data given some noise as the latent seed variable, and a discriminator network that is trained to distinguish between the generator's fake output and real data sample. The critical issue in the GAN training process is to measure the difference between the real data distribution and the generated fake data distribution. For better measurement between the two distributions, Wasserstein GANs [4] are proposed with better training stability over prior Jensen-Shannon (JS) divergence and Kullback-Leibler (KL) divergence GANs [46].

Wasserstein GANs have attracted much interest in the community recently. The original Wasserstein GAN and its variants employ Kantorovich-Rubinstein duality, which involves a saddle-point objective with a strong 1-Lipschitz constraint for the discriminator function. On the other hand, some researchers approximate the Wasserstein distance in the primal form rather than solving a dual problem to improve stability. [33] introduces a random projection to estimate the Wasserstein distance from samples directly. [97] employs the Sinkhorn algorithm to calculate the primal form of Wasserstein distance with entropic regularization. Although the performance of the state-of-the-art Wasserstein GAN is better than the original version, the computation complexity of the approximation to dual or primal Wasserstein measures keeps increasing.

In general, the Wasserstein GAN framework consists of two time-consuming components: the forward and backward updates of the discriminator and the generator, and the approximation of Wasserstein distance. Since the discriminator employs the approximated Wasserstein distance to estimate the distance between the real distribution and the generated distribution, we should guarantee the correctness of the Wasserstein distance per generator update. Otherwise, the discriminator may mislead the generator during the training process. Overall, the computation cost is much heavier when the inaccurate approximation involves the frequent update of the discriminator. In practice, we usually spend several days training a generator model with edging NVIDIA GPUs.

In this chapter, we explore the parallel computation to speed up the Wasserstein GAN training. A straightforward solution is based on data-parallelism. A set of worker unit containing model replicas is fed with different partitions of the input dataset, and the model replicas are synchronized via a parameter server. Unfortunately, the above approach is not ideal because of the skew parameter updates in GAN framework—most of GAN algorithms update the discriminator several times for every generator update. The frequent update of discriminator easily becomes a bottleneck due to the limited communication bandwidth in all distributed systems.

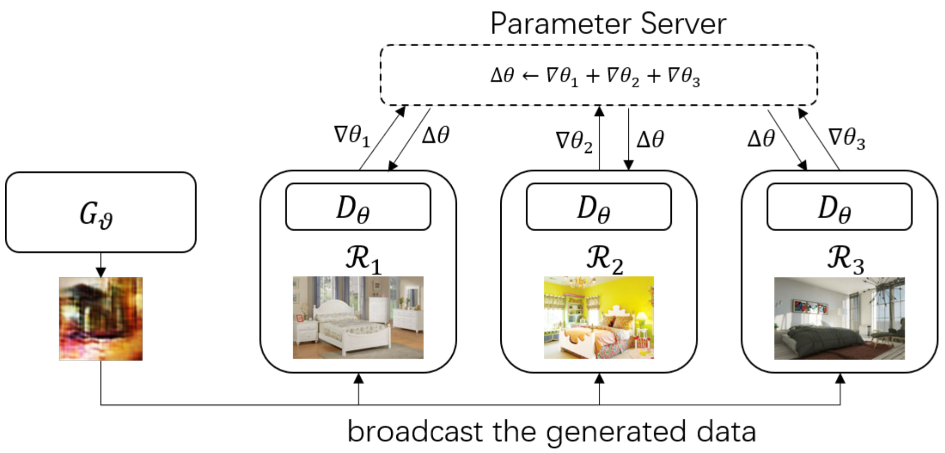
In order to eliminate the communication bottleneck for the synchronization of the discriminator, we intend to remove the synchronization requirements by allowing the discriminators different among worker units. Besides this, we also follow the

data-parallelism to split the whole dataset into different worker unit. In this case, we can consider that each worker unit contains different real distribution supported by portions of the whole dataset. Then we formulate the objective function as minimizing the averaged Wasserstein distance between the generated distribution and the multiple real distributions. Inspired by [43, 108], we optimize the objective function by the stochastic algorithm for discrete optimal transport problem, which can be efficiently paralleled.
In summary, our contributions to parallel Wasserstein GANs are highlighted as follows:

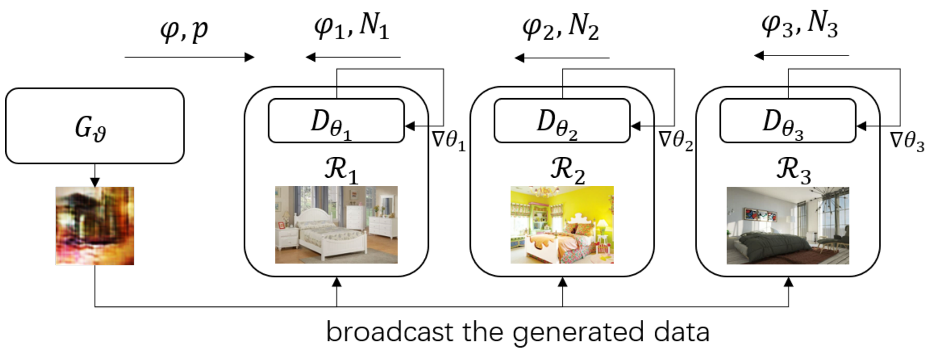
- To our best knowledge, we are the first to propose a parallel architecture for Wasserstein GANs framework to speed up the training of GANs from the computational perspective.
- Different from common Wasserstein GANs, we develop an efficient stochastic algorithm to approximate the Wasserstein distance with higher accuracy for the newly proposed parallel framework.
- In experiments, we show that the proposed parallel architecture enjoys the superior convergence speed and comparable stability.

6.2 Methodology

In all Wasserstein GANs framework, the Wasserstein distance plays a pivotal role to estimate the distance between the real distribution $\mathbb{P}_r$ and the generated distribution $\mathbb{P}_g$. In real-world applications, we estimate the distributions by the empirical probabilistic measure with sampled data. Formally,



(a) Parameter Server



(b) Proposed Architecture

Figure 6.1: Comparisons of the popular parameter server and the proposed architecture.

we formulate the empirical measure of the real distribution μ_r and the artificially generated distribution μ_g by Dirac mass as follows:

$$\mu_r = \sum_{i=1,x\in\mathcal{R}}^{n_r} p_i^r \delta_{x_i^r}, \tag{6.1}$$

$$\mu_g = \sum_{j=1,x\in\mathcal{G}}^{n_g} p_j^g \delta_{x_j^g}, \tag{6.2}$$

where $\delta_{x_i^r}$ and $\delta_{x_j^g}$ are Dirac function at location x_i^r and x_j^g, p_i^r and p_j^g are probability masses associated to the i-th real and j-th generated samples respectively, and n_r and n_g are the cardinality of the set of real data and generated fake data in a mini-batch respectively.

When the number of data in each mini-batch increases, the empirical distribution μ_r approximates the real distribution $\mathbb{P}_r$ with higher accuracy [42]. However, the size of the mini-batch is constrained due to the limited memory in each worker (e.g., GPU). To solve this problem, we turn to a parallel training mechanism in a distributed environment.

6.2.1 Framework

We propose a communication-efficient framework for parallel Wasserstein GAN. The details are shown in Figure 6.1.

Figure 6.1 (a) shows the traditional parallel approach [31, 70]. In this framework, we split the training dataset across several workers first. Then, for each iteration, each worker with a model replica conducts forward-backward computation to calculate the gradients in parallel. Finally, the model parameters are synchronized with a global parameter server before the next iteration starts. This framework is widely used in distributed deep learning systems like Tensorflow [1].

However, the traditional parallel approach in Figure 6.1 (a) contains two drawbacks. The parameter server is not proper to train the discriminator for GAN framework. Because the frequent synchronization of the parameters in the discriminator and the generator leads to massive communication demand, which degrades the training speed in any parallel solutions. Another drawback of the parameter server comes from the gradient aggregation for a large-scale batch. Traditionally, the gradient used for back-propagation is the arithmetic mean of gradients from different portions of mini-batch. The averaged gradient can not preserve the geometry information, which is vital to depict the semantic information.

To overcome the drawbacks in traditional parallel approaches, we propose a new parallel framework for Wasserstein GAN training, shown in Figure 6.1 (b). In our approach, we assign K worker unit with a discriminator D_{θ_k} and a master unit with a generator G_{ϑ}. θ_k and ϑ represent the parameters of the neural networks in the discriminator k and the generator respectively. Note that $\theta_k \neq \theta_{k'}$ in our proposed architecture generally. Our proposed framework relaxes the communication constraints by eliminating the synchronization of the discriminator among workers. In addition, this relaxed requirement could improve the diversity of the discriminator model and intuitively enhance the generator by increasing the difficulty of cheating the discriminator.

In our proposed architecture, the training real dataset $\mathcal{R}$ is split into K workers. Then, for worker k, the empirical measure μ_k^r is supported by the subset of real data $\mathcal{R}_k$.

Similar to the original Wasserstein GAN, we try to minimize the sum of the Wasserstein distance between the generated distribution μ_g and all real distribution μ_k^r with empirical measures as follows:

$$\mathcal{L}\left(\mu_g, \{\mu_k^r\}_{k=1}^K\right) = \frac{1}{K}\sum_{k=1}^{K} W_c\left(\mu_g, \mu_k^r\right). \tag{6.3}$$

6.2.2 Parallel Wasserstein GAN Algorithm

We develop a novel parallel Wasserstein GAN algorithm to solve the objective function in the our proposed architecture, namely Eq. (6.3). The loss function defined in Eq. (6.3) is similar to the Wasserstein barycenter problem introduced in [3]. Intuitively, the optimal generated fake distribution should be approximated to the barycenter of the real distributions in all workers. Specifically, we employ the empirical distribution of the barycenter of all real distributions as the supervised information to train the generator network. In order to solve the problem in Eq. (6.3) with more accuracy, we consider to calculate the Wasserstein distance directly, which is different from the recently popular solutions with entropic regularization [43, 27]. In the proposed framework, we follow [43, 25] to consider the semi-discrete Kantorovich's dual formulations of the primal optimal transport problem in Eq. (2.22):

$$\begin{aligned} &W_c\left(\mu_g, \mu_r\right) = \\ &\max_{\varphi \in \mathbb{R}^{n_g}} \left\{ \sum_{j=1}^{n_g} \varphi_j p_j^g + \int_\Omega \varphi^c(x_r) \mathrm{d}\mu_r(x_r) \right\}, \end{aligned} \tag{6.4}$$

where $\varphi^c\left(x_r\right) = c\left(x_r, x_j^g\right) - \varphi_j$ is the c-transform of the Kantorovich potential φ_j [116]. Therefore, $\mathcal{L}$ is reformulated as:

ALGORITHM 5: Generator as Master Unit

```
Input: $K$: number of worker unit, $n_g$: batch size in generator, $n_r$:
       batch size in worker, $T$: number of iteration for the estimation
       of the probability Dirac mass.
Output: $G_\vartheta$: the generator model.
while $\vartheta$ has not converged do
    Sample Gaussian noise $z_1, \dots, z_{n_g}$ from $\mathcal{N}(0,1)$
    Generate fake images $\left\{x_j^g; x_j^g = G_\vartheta\left(z_j\right)\right\}_{j=1}^{n_g}$
    $\varphi_j \leftarrow 0 \; \forall j \in [1, n_r]$
    Broadcast $\{x_j^g\}_{j=1}^{n_g}$ to workers
    for $t$=1,...,$T$ do
        $\varphi_j \leftarrow \frac{1}{K}\sum_{k=1}^{K} \varphi_j^k \; \forall j \in [1, n_g]$
        Broadcast $\{\varphi_j, p_j^g\}_{j=1}^{n_g}$ to workers
        Receive $\{\varphi_j^k, N_j^k\}_{j=1}^{n_g}$ from workers
        $p_j^g \leftarrow \frac{1}{K}\sum_{k=1}^{K} N_j^k \; \forall j \in [1, n_g]$
    end
    Conduct back-propagation with the loss:
    Loss($\{D_\theta^g(G_\vartheta(z_j))\}_{j=1}^{n_g}, \{p_j^g\}$)
end
```

$$\begin{aligned}&\mathcal{L}\left(\mu_g, \{\mu_k^r\}_{k=1}^K\right) = \\ &\max_{\varphi\in\mathbb{R}^{n_g}} \left\{ F(\varphi) = \sum_{j=1}^{n_g} \varphi_j p_j^g + \frac{1}{K}\sum_{k=1}^K \int_{\Omega_k} \varphi^c(x_r) \mathrm{d}\mu_k^r(x_r) \right\}\end{aligned} \tag{6.5}$$

From Eq. (6.5), we can know that $F(\varphi)$ is a concave function. Then we can derive W_c from the optimization using the stochastic gradient ascent method. Since $F(\varphi)$ is derivative with respective to each individual φ_j:

$$\frac{\partial F}{\partial \varphi_j} = p_j^g - \frac{1}{K}\sum_{k=1}^K \int_{\mathrm{Vor}_{\varphi_j}^k} \mathrm{d}\mu_k^r\left(x_r\right), \tag{6.6}$$

where Vor_{φ_j} is Voronoi cells of the point x_j^g [98] defined in the following:

$$\begin{aligned}&\mathrm{Vor}_{\varphi_j} = \\ &\left\{x \in \mathcal{R} \ : \ c\left(x, x_j^g\right) - \varphi_j \leq c\left(x, x_{j'}^g\right) - \varphi_{j'} \forall j'\right\}.\end{aligned} \tag{6.7}$$

Therefore, W_c is obtained when $\frac{\partial F}{\partial \varphi_j} = 0$ and the optimal p_j^* is calculated as:

$$\begin{aligned} p_j^* &= \frac{1}{K}\sum_{k=1}^K \int_{\mathrm{Vor}_{\varphi_j}^k} \mathrm{d}\mu_k^r\left(x_r\right) && (6.8)\\ &= \frac{1}{K}\sum_{k=1}^K \mathbb{E}_{x^r\sim\mu_k^r}\left[I_{x^r\in\mathrm{Vor}_{\varphi_j}^k}\right] && (6.9)\\ &\approx \frac{1}{K}\sum_{k=1}^K \sum_{i=1}^{n_k^r} I_{x_i^k\in\mathrm{Vor}_{\varphi_j}^k}, && (6.10)\end{aligned}$$

where I is denoted as the indicator function of a given set. p_j^* is the valuable supervised information of our GAN framework for the generated fake point x_j^g since p_j^g represents the weight of the point x_j^g in the empirical distribution.

In our framework, the computation of p_j^* in Eq. (6.8) is easily decomposed to all workers in parallel, because the Voronoi cells $\mathrm{Vor}_{\varphi_j}^k$ in each worker k are independent with each other. When the empirical estimation in each worker is finished, p_j^* is reduced to the master worker with x_j^g. Algorithm 5 and 6 detail the proposed parallel algorithm with mini-batch.

ALGORITHM 6: Discriminator as Worker Unit

```
Input: $K$: number of worker unit, $n_r$: batch size in worker unit, $\eta$ :
       learning rate for Kantorovich potential, $T$: number of
       iterations for the Dirac mass estimation.
forall worker $k \in [1, K]$ do
    Sample $\{x_i\}_{i=1}^{n_r}$ a batch from the real dataset $\mathcal{R}_k$
    Receive fake images $\left\{x_j^g\right\}_{j=1}^{n_g}$ from the master unit
    Evaluate cost $c_{ij} = \left\| D_{\theta_k}(x_i^r) - D_{\theta_k}\left(x_j^g\right) \right\| \ \forall i \in [1, n_r], j \in [1, n_g]$
    for $t = 1, \dots, T$ do
        Receive $\{\varphi_j, p_j^g\}_{j=1}^{n_g}$ from master
        for $j = 1, \dots, n_g$ do
            Compute $\mathrm{Vor}_{\varphi_j}$ from Eq. (6.7)
            $N_j^k \leftarrow \sum_{i=1}^{n_k^r} I_{x_i^k \in \mathrm{Vor}_{\varphi_j}^k}$
            $\varphi_j^k \leftarrow \varphi_j - \eta N_j^k$
        end
        Send $\{\varphi_j^k, N_j^k\}_{j=1}^{n_g}$ to master
    end
    Conduct back-propagation with the loss:
    $\mathrm{Loss}(\{D_\theta^k(x_j^g)\}_{j=1}^{n_g}, \{N_j^k\})$
end
```

In many machine learning algorithms armed with Wasserstein distance, the ground cost function on the given metric is relatively simple, such as the ℓ_2 norm in Euclidean space. However, ℓ_2 norm is not proper for high-dimensional image data in GAN framework, because it lacks complex geometric information. Therefore, we parameterize the cost function $c\left(x_r, x_g\right)$ from the discriminator as follows:

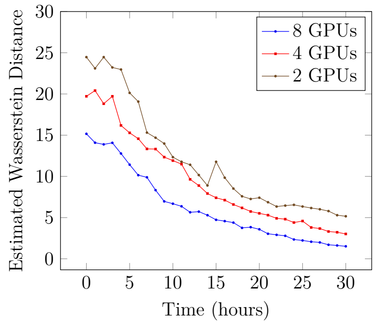
$$c_\theta\left(x_r, x_g\right) = \left\|D_\theta\left(x_r\right) - D_\theta\left(x_g\right)\right\|, \tag{6.11}$$

where D_θ is the discriminator network with parameters θ. When p^* is obtained, we update the discriminator network D_θ^k in all workers and the generator network G_ϑ by back-propagation.

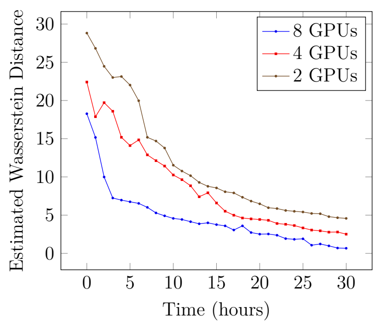
It is important to note that our approach does not contain any cumbersome communication policy that may increase the negative effect of mismatching communication bandwidth in the distribution environment. For each worker unit, they only receive generated data and send a simple vector of size n_g to optimize the proposed objective function collaboratively. Moreover, our approach is compatible with different configurations for parallel computation: (a) All GPUs are installed via PCI Express in a single workstation. In this setting, each GPU is considered as an isolated computation worker in the parallel concept. (b) There are multiple machines, each of which may contain more than one GPU, deemed as distributed computation. In the following, we take experiments in case (b) to evaluate our approach.

6.3 Evaluations

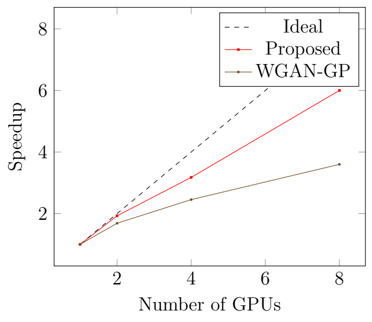
In the experimental part, we evaluate the proposed parallel approach on two widely used image datasets CIFAR-10 and LSUN [133]. CIFAR-10 contains 60,000 small color images with size 32×32. LSUN is a large-scale scene understanding dataset. Following the setting in many Wasserstein GANs papers, we



(a) WGAN-GP with default distributed framework



(b) Proposed



(c) Speedup

Figure 6.2: The estimated Wasserstein distance of WGAN-GP and the proposed method both trained on LSUN Bedroom dataset. To make fair comparisons, we employ PyTorch's default MPI interface to train WGAN-GP in parallel. In (c), we calculate the speedup based on the execution time from the start to the first checkpoint where the estimated Wasserstein distance is less than 5.0.

mainly use the bedroom category from LSUN dataset. The bedroom category contains 3 million color images with a size of 128×128.

All experiments in this section are conducted in a cluster with four machines with 2 NVIDIA GTX 1080 GPUs each. Therefore, we could scale our approach up to 8 GPUs setting, which contains one master unit for the generator and seven workers for the discriminators.

We implement our algorithms with PyTorch [88], and utilize the 101-layer ResNet block in generator and discriminator. We assign the number of iteration for per mini-batch update $T = 5$ for all experiments. Intuitively, the probability of Dirac mass for each generator update should be more accurate when T increases. However, larger T means that the communication rounds between the master and workers will increase as well.

6.3.1 Convergence Speed for Scalability

In a parallel environment, we assume that all workers have the same batch size n_r. Therefore, in the proposed parallel approach, the total number of real images sampled from the dataset for a single generator update is $K \cdot n_r$, where K is the number of worker unit. Usually, the range of n_r is constrained to the memory in GPUs. Then, the batch size of real images scales linearly with the number of machines. For the special "2 GPUs" setting in experiments, we utilize two GPUs equipped at different machines to make a fair comparison, because the inner-machine communication is much faster than the inter-machine protocol.

To demonstrate the advantage of our proposed method compared to the traditional distributed deep learning framework, we implement parallel WGAN-GP through MPI interface in PyTorch. MPI follows All-Reduce policy, equivalent to parameter server framework depicted in Figure 6.1 (a).

Figure 6.2 (a) and (b) illustrate the convergence speed with regard to the number of the GPUs, where the y-axis shows the change of estimated Wasserstein distance between the real distribution and the generated distribution during the training. From the figures, we can see both of the proposed method and WGAN-GP with more GPUs always converge faster than the same approach with fewer GPUs.

We also employ the standard speedup measurement in distributed computation to demonstrate the scalability of our proposed method, and the measurement is defined as below:

$$\text{Speedup}(N) = \frac{\text{The execution time of one unit}}{\text{The execution time of } N \text{ units}}. \tag{6.12}$$

In the experiments, we measure the duration from starts to the first checkpoint where the corresponding estimated Wasserstein distance is less than 5.0. We consider 5.0 as an ending point because all methods converge with less vibration when the estimated distance is less than 5.0. From Figure 6.2 (c), we can easily conclude that our method enjoys a better convergence speed than the traditional distributed deep learning framework, which is coincident with our expectation.

Due to limited space, we do not demonstrate the figures for CIFAR10 dataset. Actually, the behavior of two methods in CIFAR10 is quite similar to Figure 6.2–both methods converge faster if more GPUs, and our approach is faster than the traditional parallel framework.

6.3.2 Quantitative and Qualitative Evaluation

We first conduct the quantitative evaluation to demonstrate the stability of the proposed method. We calculate the Inception Score during the training and evaluate the generator performance on CIFAR-10. A higher Inception Score indicates a better ability of the generative model to produce samples with variability. Practically, we calculate the maximal Inception Score reached in 20,000 generator updates.

We also follow [126] to evaluate the model performance on LSUN dataset by Fréchet Inception Distance (FID) [53], which measures the difference between real and fake data distributions.

Table 6.1: Quantitative evaluations on CIFAR-10 and LSUN dataset. Smaller FID is better, larger IS is better.

Method	LSUN bed. (FID)	CIFAR-10 (IS)
WGAN-GP (8 GPUs)	27.3	7.73
Ours (2 GPUs)	23.2	7.12
Ours (4 GPUs)	21.9	7.68
Ours (8 GPUs)	**21.0**	**7.81**

From Table 6.1, we observe that the performance of the proposed method becomes better when the number of worker unit increases, which mainly benefits from larger size of real data batch among multiple worker units. Moreover, our method enjoys much better performance over WGAN-GP.

We also take qualitative evaluation from the visualization performance. Figure 6.3 indicates that the generated images of our proposals with different number of GPUs are comparable to results in WGAN-GP. Moreover, comparable results in Figure 6.3 (b-d) show that our parallel method has no restriction on the number of GPUs used to generate plausible results. Figure 6.4 shows the performance of our proposed method is also comparable with WGAN-GP on CIFAR-10 data. Hence our method enjoys superior performance concerning the convergence speed with the comparable generation quality.

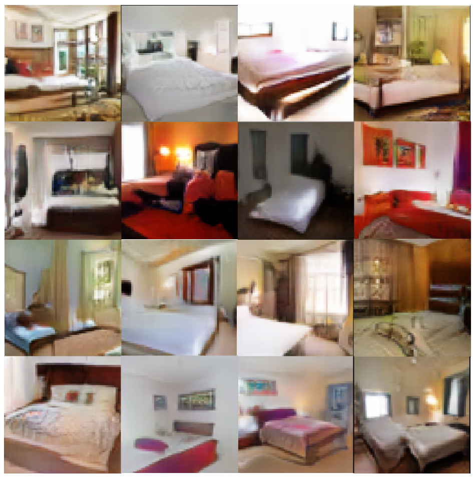
6.4 Summary

In this chapter, we introduce a novel parallel architecture designed for GANs framework to speed up the costly computation in Wasserstein GANs with multiple computation units. Experimental reports demonstrate that our proposed architecture enjoys an attractive scalability performance to decrease the training time remarkably. Moreover, both quantitative scores and qualitative demonstration display the proposed loss function with the Wasserstein distance and the parallel algorithm are reasonable and suitable for the parallel environment.

□ **End of chapter.**

(a) WGAN-GP (8 GPUs)

(b) Proposed (2 GPUs)


(c) Proposed (4 GPUs)

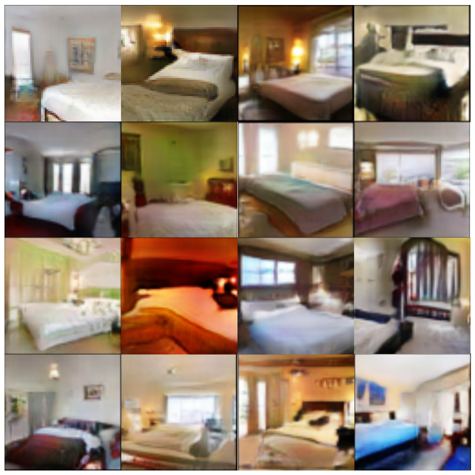
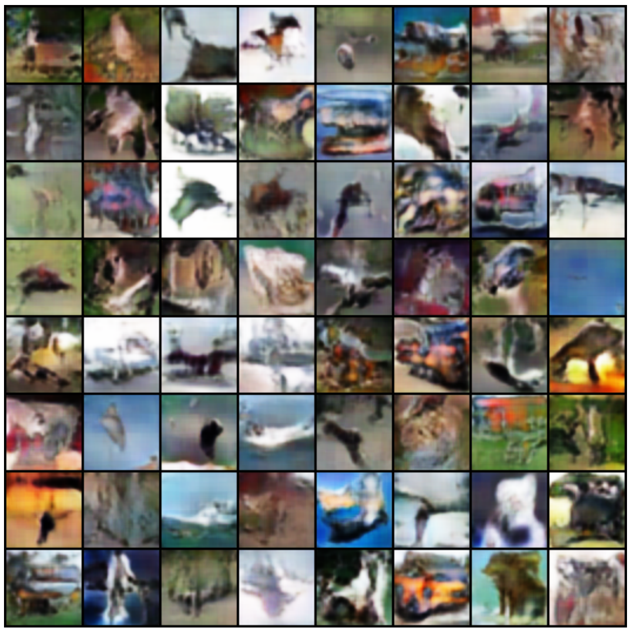

(d) Proposed (8 GPUs)

Figure 6.3: Qualitative comparisons between the state-of-the-art WGAN-GP and the proposed method with different configurations. Images in (a) are generated by WGAN-GP; images in (b, c, d) are generated by the proposed model trained with the different number of GPUs. All models are trained on LSUN bedroom dataset within 24 hours.


(a) WGAN-GP (8 GPUs)

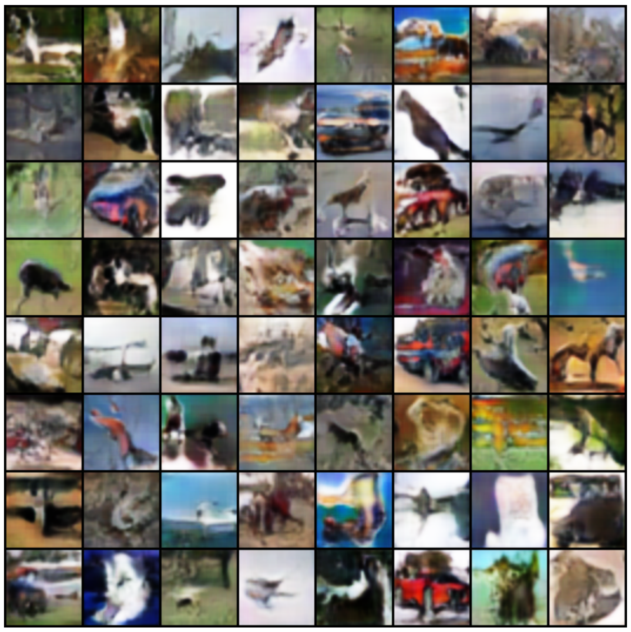

(b) Proposed (8 GPUs)

Figure 6.4: Qualitative comparisons between the state-of-the-art WGAN-GP and the proposed method.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

In this thesis, we explore distributed distance learning algorithms from a variety of perspectives, including global distance learning, local distance learning, deep distance learning, and probabilistic distance learning.

In chapter 3, we have developed a parallel global distance learning algorithm from the popular Information-Theoretic Metric Learning in order to learn the proper distance function from high-dimensional data without low-rank approximation, which is infeasible for previous distance learning algorithms.

Moreover importantly, we present a rigorous theoretical analysis to upper bound the Bregman divergence between the sequential algorithm and the parallel algorithm, which guarantees the correctness and performance of the proposed algorithm.

In chapter 4, we focus on improving the performance of local distance learning framework with a specific application called learning to rank. Specifically, we firstly developed a localized GMML algorithm to adopt distance learning for ranking algorithm from a query-dependent model to widely used query-independent model. Finally, we have demonstrated empirical experiments to prove that our approach outperforms both of the state-of-the-art query-dependent algorithms and query-independent algorithms.

In chapter 5, we present a scalable solution to tackle the big data challenge for deep distance learning algorithm by mining hard-negative samples in a distributed environment. Furthermore, we demonstrate a hybrid synchronization policy to speed up the training process of deep distance learning algorithm by reducing the communication workload significantly in a distributed environment.

In chapter 6, we introduce a novel parallel architecture to accelerate the costly computation in Wasserstein GANs in a distributed environment. We further demonstrate that our proposed architecture enjoys an attractive scalability performance to decrease the training time remarkably.

7.2 Future Work

Distance learning algorithms have been widely studied in the recent two decades. Distributed computation strategy is a promising research direction to solve big data and big model challenges in distance learning. Although there are plenty of well-designed distance learning algorithms and distributed optimization methods in the literature respectively, the joint design is still not adequately explored.

- **Distributed global distance learning**

  Global distance learning mainly involves the heavy computation of solving semi-definite program (SDP). The convex combination of rank-one matrices introduced in Chapter 3 with special structure or regularization is a proper tool to address the SDP problem without strictly constraints about matrix rank.

The popular Frank-Wolfe [58] and its variants solve linear minimization oracles as a subroutine to optimize the rank-one matrices combination. Therefore, the parallel block Frank-Wolfe algorithm [121] is expected to enhance the scalability performance of parallel global distance learning algorithm.

- **Distributed local distance learning**

  Most local distance learning algorithms address the high dimensional problem by projecting the original data into local dimensional embedding. Usually, any direct projection-based optimization techniques involving eigendecomposition is too expensive in time complexity. In optimization theory, we could assume the low dimensional embedding lies on the surface of a Riemannian manifold to reduce the time complexity of the projection operators. Therefore, the parallel method of Riemannian manifold optimization should be explored.

- **Distributed deep distance learning**

  The key challenge of deep distance learning is the hard example mining in each mini-batch during the training process. Existing methods focusing on hand-crafted sampling policy is limited in the convergence speed. Hence, to accelerate the convergence of deep distance learning, a smarter and dynamic semi-hard negative example mining in a distributed environment is valuable to explore.

- **Distributed Wasserstein distance**

  The accurate computation of Wasserstein distance enjoys heavy computation. Any straightforward distributed approach can easily meet the bottleneck of the scalability issue due to the complex structure of the original model. To reduce the complexity of the original models and algorithms, regularized or sliced variants of Wasserstein distance is taken as a trade-off between the time complexity and model performance. Therefore, the problem of how to take advantage of distributed optimization for regularized constraints is a promising research direction to explore.

□ **End of chapter.**

Chapter 8

Publications during Ph.D. Study

1. **Yuxin Su**, Shenglin Zhao, Xixian Chen, Irwin King and Michael Lyu. “Parallel Wasserstein Generative Adversarial Nets with Multiple Discriminators" In Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI), 2019.

2. **Yuxin Su**, Michael Lyu, and Irwin King. “Communication Efficient Distributed Deep Metric Learning with Hybrid Synchronization." In Proceedings of the 27th ACM International Conference on Information and Knowledge Management (CIKM), 2018.

3. **Yuxin Su**, Irwin King, and Michael Lyu. “Learning to Rank Using Localized Geometric Mean Metrics." In Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR), 2017.

4. **Yuxin Su**, Haiqin Yang, Irwin King, and Michael Lyu. “Distributed Information Theoretic Metric Learning in

Apache Spark." In 2016 International Joint Conference on Neural Networks (IJCNN), 2016.

5. Haiqin Yang, Guang Ling, **Yuxin Su**, Michael R. Lyu, and Irwin King. "Boosting response aware model-based collaborative filtering." IEEE Transactions on Knowledge and Data Engineering 27, no. 8 (2015): 2064-2077.

6. **Yuxin Su**, Haiqin Yang, Michael Lyu, Irwin King. "Distributed Non-negative Matrix Factorization with Loose Synchronization." In Proceeding of the WSDM workshop on Scalable Data Analytics, 2015

7. Junjie Hu, Haiqin Yang, **Yuxin Su**, Michael Lyu, Irwin King. "Accelerated Information-Theoretic Metric Learning." In Proceeding of the WSDM workshop on Scalable Data Analytics, 2015

Note: The papers [1, 2, 3, 4] are partially involved in this thesis.

□ **End of chapter.**

Bibliography

[1] M. Abadi, P. Barham, et al. Tensorflow: A system for large-scale machine learning. In *OSDI*, pages 265–283, 2016.

[2] A. Agarwal and J. C. Duchi. Distributed delayed stochastic optimization. In *Advances in Neural Information Processing Systems*, pages 873–881, 2011.

[3] M. Agueh and G. Carlier. Barycenters in the wasserstein space. *SIAM J. Math. Analysis*, 43(2):904–924, 2011.

[4] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein generative adversarial networks. In *ICML*, pages 214–223, 2017.

[5] S. Bell and K. Bala. Learning visual similarity for product design with convolutional neural networks. *ACM Trans. Graph.*, 34(4):98:1–98:10, 2015.

[6] A. Bellet, A. Habrard, and M. Sebban. A survey on metric learning for feature vectors and structured data. *arXiv preprint arXiv:1306.6709*, 2013.

[7] A. Bellet, A. Habrard, and M. Sebban. A survey on metric learning for feature vectors and structured data. *arXiv preprint arXiv:1306.6709*, 2013.

[8] O. Bousquet, S. Gelly, I. Tolstikhin, C.-J. Simon-Gabriel, and B. Schoelkopf. From optimal transport to generative modeling: the vegan cookbook. *arXiv preprint arXiv:1705.07642*, 2017.

[9] J. K. Bradley, A. Kyrola, D. Bickson, and C. Guestrin. Parallel coordinate descent for l1-regularized loss minimization. *arXiv preprint arXiv:1105.5379*, 2011.

[10] L. M. Bregman. The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming. *USSR computational mathematics and mathematical physics*, 7(3):200–217, 1967.

[11] J. Bromley, J. W. Bentz, L. Bottou, I. Guyon, Y. LeCun, C. Moore, E. Säckinger, and R. Shah. Signature verification using A "siamese" time delay neural network. *IJPRAI*, 7(4):669–688, 1993.

[12] D. Broomhead, D. S. and Lowe. Multivariable Functional Interpolation and Adaptive Networks. *Complex Systems*, 2:321– 355, 1988.

[13] M. Bucher, S. Herbin, and F. Jurie. Improving semantic embedding consistency by metric learning for zero-shot classiffication. In *Computer Vision - ECCV 2016*, pages 730–746, 2016.

[14] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. In *Proceedings of the 22nd international conference on Machine learning*, pages 89–96, 2005.

[15] C. J. C. Burges. From RankNet to LambdaRank to LambdaMART : An Overview. *Technical Report*, 2010.

[16] C. J. C. Burges, K. M. Svore, P. N. Bennett, A. Pastusiak, and Q. Wu. Learning to Rank Using an Ensemble of Lambda-Gradient Models. *Journal of Machine Learning Research (JMLR): Workshop and Conference Proceedings*, 14:25–35, 2011.

[17] M. Campo, J. Espinoza, J. Rieger, and A. Taliyan. Collaborative metric learning recommendation system: Application to theatrical movie releases. *arXiv preprint arXiv:1803.00202*, 2018.

[18] Z. Cao, T. Qin, T.-Y. Liu, M.-F. Tsai, and H. Li. Learning to rank. In *Proceedings of the 24th international conference on Machine learning - ICML '07*, pages 129–136, New York, New York, USA, 6 2007. ACM Press.

[19] O. Chapelle. Yahoo ! Learning to Rank Challenge Overview. *JMLR: Workshop and Conference Proceedings*, 14:1–24, 2011.

[20] T. Chavdarova and F. Fleuret. Sgan: An alternative training of generative adversarial networks. In *CVPR*, 2018.

[21] G. Chechik, V. Sharma, U. Shalit, and S. Bengio. Large Scale Online Learning of Image Similarity Through Ranking. *The Journal of Machine Learning Research*, 11:1109–1135, 3 2010.

[22] A. I. Chen and A. E. Ozdaglar. A fast distributed proximal-gradient method. In *2012 50th Annual Allerton*

Conference on Communication, Control, and Computing, Allerton Park & Retreat Center, Monticello, IL, USA, October 1-5, 2012, pages 601–608, 2012.

[23] J. Chen, T. Yang, and S. Zhu. Efficient low-rank stochastic gradient descent methods for solving semidefinite programs. In *AISTATS*, pages 122–130, 2014.

[24] T. Chen, M. Li, Y. Li, M. Lin, N. Wang, M. Wang, T. Xiao, B. Xu, C. Zhang, and Z. Zhang. Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems. *CoRR*, abs/1512.01274, 2015.

[25] S. Claici, E. Chien, and J. Solomon. Stochastic wasserstein barycenters. In *ICML*, pages 998–1007, 2018.

[26] Y. Cui, F. Zhou, Y. Lin, and S. J. Belongie. Fine-grained categorization and dataset bootstrapping using deep metric learning with humans in the loop. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1153–1162, 2016.

[27] M. Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In *NeurIPS*, pages 2292–2300, 2013.

[28] J. V. Davis and I. S. Dhillon. Structured metric learning for high dimensional problems. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Las Vegas, Nevada, USA, August 24-27, 2008*, pages 195–203, 2008.

[29] J. V. Davis, B. Kulis, P. Jain, S. Sra, and I. S. Dhillon. Information-theoretic metric learning. In *Proceedings of the 24th international conference on Machine learning*,

pages 209–216, New York, New York, USA, 6 2007. ACM Press.

[30] C. C. de Sá, M. A. Gonçalves, D. X. Sousa, and T. Salles. Generalized BROOF-L2R. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 95–104, New York, New York, USA, 7 2016. ACM Press.

[31] J. Dean and G. C. et al. Large scale distributed deep networks. In *NeurIPS*, pages 1232–1240, 2012.

[32] M. Der and L. K. Saul. Latent coincidence analysis: A hidden variable model for distance metric learning. In *Advances in Neural Information Processing Systems*, pages 3230–3238, 2012.

[33] I. Deshpande, Z. Zhang, and A. G. Schwing. Generative modeling using the sliced wasserstein distance. In *CVPR*, June 2018.

[34] J. Dhillon. Differential entropic clustering of multivariate gaussians. *Advances in Neural Information Processing Systems*, 19:337, 2007.

[35] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, pages 647–655, 2014.

[36] E. Fetaya and S. Ullman. Learning local invariant mahalanobis distances. In *Proceedings of the 32nd Inter-*

national Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015, pages 162–168, 2015.

[37] M. Frank and P. Wolfe. An algorithm for quadratic programming. *Naval research logistics quarterly*, 3(1-2):95–110, 1956.

[38] Y. Freund, R. Iyer, R. E. Schapire, and Y. Singer. An Efficient Boosting Algorithm for Combining Preferences. *Journal of Machine Learning Research*, 4:933–969, 2003.

[39] B. J. Frey and D. Dueck. Clustering by passing messages between data points. *science*, 315(5814):972–976, 2007.

[40] J. H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29(5):1189–1232, 2001.

[41] Y. Ganjisaffar, R. Caruana, and C. V. Lopes. Bagging gradient-boosted trees for high precision, low variance ranking models. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information - SIGIR '11*, page 85, New York, New York, USA, jul 2011. ACM Press.

[42] A. Genevay, L. Chizat, F. Bach, M. Cuturi, and G. Peyré. Sample complexity of sinkhorn divergences. *arXiv preprint arXiv:1810.02733*, 2018.

[43] A. Genevay, M. Cuturi, G. Peyré, and F. R. Bach. Stochastic optimization for large-scale optimal transport. In *NeurIPS*, pages 3432–3440, 2016.

[44] A. Genevay, G. Peyré, and M. Cuturi. Learning generative models with sinkhorn divergences. In *AISTATS*, pages 1608–1617, 2018.

[45] A. Globerson and S. T. Roweis. Metric learning by collapsing classes. In *Advances in Neural Information Processing Systems 18 [Neural Information Processing Systems, NIPS 2005, December 5-8, 2005, Vancouver, British Columbia, Canada]*, pages 451–458, 2005.

[46] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, and Y. Bengio. Generative adversarial nets. In *NeurIPS*, pages 2672–2680, 2014.

[47] P. Goyal, P. Dollár, R. B. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, and K. He. Accurate, large minibatch SGD: training imagenet in 1 hour. *CoRR*, abs/1706.02677, 2017.

[48] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville. Improved training of wasserstein gans. In *NeurIPS*, pages 5769–5779, 2017.

[49] R. Hadsell, S. Chopra, and Y. LeCun. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1735–1742, 2006.

[50] M. Harandi. Siamese networks : A thing or two to know. 2017.

[51] S. Hauberg, O. Freifeld, and M. J. Black. A Geometric take on Metric Learning. In *Advances in Neural Information Processing Systems*, pages 2024–2032, 2012.

[52] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[53] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *NeurIPS*, pages 6629–6640, 2017.

[54] E. Hoffer and N. Ailon. Semi-supervised deep learning by metric embedding. *CoRR*, abs/1611.01449, 2016.

[55] S. C. Hoi, W. Liu, and S.-F. Chang. Semi-supervised distance metric learning for collaborative image retrieval and clustering. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 6(3):18, 2010.

[56] C. Hsieh, L. Yang, Y. Cui, T. Lin, S. J. Belongie, and D. Estrin. Collaborative metric learning. In *WWW*, pages 193–201, 2017.

[57] B. Iannazzo. The geometric mean of two matrices from a computational viewpoint. *Numerical Linear Algebra with Applications*, 23(2):208–229, 3 2016.

[58] M. Jaggi. Revisiting frank-wolfe: Projection-free sparse convex optimization. In *Proceedings of the 30th Interna-*

tional Conference on Machine Learning (ICML-13), pages 427–435, 2013.

[59] K. Järvelin and J. Kekäläinen. Cumulated gain-based evaluation of IR techniques. *ACM Trans. Inf. Syst.*, 20(4):422–446, 2002.

[60] N. Jiang, W. Liu, and Y. Wu. Order determination and sparsity-regularized metric learning adaptive visual tracking. In *2012 IEEE Conference on Computer Vision and Pattern Recognition, Providence, RI, USA, June 16-21, 2012*, pages 1956–1963, 2012.

[61] T. Kato and N. Nagano. Metric learning for enzyme active-site search. *Bioinformatics*, 26(21):2698–2704, 2010.

[62] G. Koch, R. Zemel, and R. Salakhutdinov. Siamese neural networks for one-shot image recognition. In *ICML Deep Learning Workshop*, volume 2, 2015.

[63] J. Krause, M. Stark, J. Deng, and L. Fei-Fei. 3d object representations for fine-grained categorization. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 554–561, 2013.

[64] B. Kulis. Metric Learning: A Survey. *Foundations and Trends® in Machine Learning*, 5(4):287–364, 2013.

[65] B. Kulis, P. Jain, and K. Grauman. Fast similarity search for learned metrics. *IEEE Trans. Pattern Anal. Mach. Intell.*, 31(12):2143–2157, 2009.

[66] B. Kulis, M. A. Sustik, and I. S. Dhillon. Low-rank kernel learning with bregman matrix divergences. *Journal of Machine Learning Research*, 10:341–376, 2009.

[67] J. Langford, A. Smola, and M. Zinkevich. Slow learners are fast. *arXiv preprint arXiv:0911.0491*, 2009.

[68] M. T. Law, R. Urtasun, and R. S. Zemel. Deep spectral clustering learning. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pages 1985–1994, 2017.

[69] Y. LeCun, Y. Bengio, and G. E. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[70] J. Li, J. Cheng, Y. Zhao, F. Yang, Y. Huang, H. Chen, and R. Zhao. A comparison of general-purpose distributed systems for data processing. In *Big Data*, pages 378–383, 2016.

[71] J. Li, X. Lin, X. Rui, Y. Rui, and D. Tao. A distributed approach toward discriminative distance metric learning. *Neural Networks and Learning Systems, IEEE Transactions on*, 26(9):2111–2122, Sept 2015.

[72] M. Li, D. G. Andersen, J. W. Park, A. J. Smola, A. Ahmed, V. Josifovski, J. Long, E. J. Shekita, and B. Su. Scaling distributed machine learning with the parameter server. In *11th USENIX Symposium on Operating Systems Design and Implementation, OSDI '14, Broomfield, CO, USA, October 6-8, 2014.*, pages 583–598, 2014.

[73] M. Li, D. G. Andersen, and A. Smola. Distributed delayed proximal gradient methods. In *NIPS Workshop on Optimization for Machine Learning*, 2013.

[74] P. Li, Q. Wu, and C. J. Burges. McRank: Learning to Rank Using Multiple Classification and Gradient Boosting. In *Advances in Neural Information Processing Systems*, pages 897–904, 2008.

[75] X. Lian, C. Zhang, H. Zhang, C. Hsieh, W. Zhang, and J. Liu. Can decentralized algorithms outperform centralized algorithms? A case study for decentralized parallel stochastic gradient descent. In *NIPS*, pages 5336–5346, 2017.

[76] D. Lim and G. Lanckriet. Efficient Learning of Mahalanobis Metrics for Ranking. In *Proceedings of The 31st International Conference on Machine Learning*, pages 1980–1988, 2014.

[77] D. Lim, G. R. G. Lanckriet, and B. McFee. Robust structural metric learning. In *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*, pages 615–623, 2013.

[78] K. Liu, A. Bellet, and F. Sha. Similarity Learning for High-Dimensional Sparse Data. *Aistats*, 38:1–14, 2015.

[79] T.-Y. Liu. Learning to Rank for Information Retrieval. *Foundations and Trends® in Information Retrieval*, 3(3):225–331, 6 2009.

[80] C. Lucchese, F. M. Nardini, S. Orlando, R. Perego, and N. Tonellotto. Speeding up Document Ranking with Rank-based Features. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development*

in Information Retrieval, pages 895–898, New York, New York, USA, 8 2015. ACM Press.

[81] Y. Ma and T. Zheng. Boosted sparse non-linear distance metric learning. *CoRR*, abs/1512.03396, 2015.

[82] R. Manmatha, C. Wu, A. J. Smola, and P. Krähenbühl. Sampling matters in deep embedding learning. In *ICCV*, pages 2859–2867, 2017.

[83] B. McFee. *More like this: machine learning approaches to music similarity.* PhD thesis, UC San Diego, 2012.

[84] B. McFee and G. R. Lanckriet. Metric learning to rank. *Proceedings of the 27th International Conference on Machine Learning*, pages 775–782, 2010.

[85] T. Mensink, J. J. Verbeek, F. Perronnin, and G. Csurka. Metric learning for large scale image classification: Generalizing to new classes at near-zero cost. In *Computer Vision - ECCV 2012 - 12th European Conference on Computer Vision, Florence, Italy, October 7-13, 2012, Proceedings, Part II*, pages 488–501, 2012.

[86] P. Moritz, R. Nishihara, S. Wang, A. Tumanov, R. Liaw, E. Liang, W. Paul, M. I. Jordan, and I. Stoica. Ray: A distributed framework for emerging AI applications. *CoRR*, abs/1712.05889, 2017.

[87] Y.-K. Noh, B.-T. Zhang, and D. D. Lee. Generative local metric learning for nearest neighbor classification. *Advances in Neural Information Processing Systems*, 18:417–424, 2010.

[88] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. 2017.

[89] H. Petzka, A. Fischer, and D. Lukovnicov. On the regularization of wasserstein gans. *ICLR*, 2018.

[90] G. Qi, J. Tang, Z. Zha, T. Chua, and H. Zhang. An efficient sparse metric learning in high-dimensional space via $l+1$-penalized log-determinant regularization. In *ICML*, pages 841–848, 2009.

[91] Q. Qian, I. M. Baytas, R. Jin, A. K. Jain, and S. Zhu. Similarity learning via adaptive regression and its application to image retrieval. *CoRR*, abs/1512.01728, 2015.

[92] Q. Qian, R. Jin, J. Yi, L. Zhang, and S. Zhu. Efficient distance metric learning by adaptive sampling and mini-batch stochastic gradient descent (SGD). *Machine Learning*, 99(3):353–372, 2015.

[93] Q. Qian, R. Jin, S. Zhu, and Y. Lin. Fine-grained visual categorization via multi-stage metric learning. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 3716–3724, 2015.

[94] D. Ramanan and S. Baker. Local Distance Functions A Taxonomy, New Algorithms, and an Evaluation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(4):794–806, 2011.

[95] B. Recht, C. Re, S. Wright, and F. Niu. Hogwild: A lock-free approach to parallelizing stochastic gradient descent.

In *Advances in Neural Information Processing Systems*, pages 693–701, 2011.

[96] R. Rosales and G. Fung. Learning sparse metrics via linear programming. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '06*, page 367, New York, New York, USA, aug 2006. ACM Press.

[97] T. Salimans, H. Zhang, A. Radford, and D. N. Metaxas. Improving gans using optimal transport. *ICLR*, 2018.

[98] F. Santambrogio. Optimal transport for applied mathematicians. *Birkäuser, NY*, 55:58–63, 2015.

[99] F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 815–823, 2015.

[100] M. Schultz and T. Joachims. Learning a distance metric from relative comparisons. In *Advances in Neural Information Processing Systems*, 2003.

[101] S. Shalev-Shwartz and Y. Singer. Logarithmic regret algorithms for strongly convex repeated games. *The Hebrew University*, 2007.

[102] C. Shen, J. Kim, L. Wang, and A. van den Hengel. Positive semidefinite metric learning using boosting-like algorithms. *Journal of Machine Learning Research*, 13:1007–1036, 2012.

[103] E. Simo-Serra, E. Trulls, L. Ferraz, I. Kokkinos, P. Fua, and F. Moreno-Noguer. Discriminative learning of deep convolutional feature point descriptors. In *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*, pages 118–126, 2015.

[104] K. Sohn. Improved deep metric learning with multi-class n-pair loss objective. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 1849–1857, 2016.

[105] H. O. Song, S. Jegelka, V. Rathod, and K. Murphy. Deep metric learning via facility location. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 2206–2214, 2017.

[106] H. O. Song, Y. Xiang, S. Jegelka, and S. Savarese. Deep metric learning via lifted structured feature embedding. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 4004–4012, 2016.

[107] S. Sra. Scalable nonconvex inexact proximal splitting. In *Advances in Neural Information Processing Systems*, pages 539–547, 2012.

[108] M. Staib, S. Claici, J. M. Solomon, and S. Jegelka. Parallel streaming wasserstein barycenters. In *NeurIPS*, pages 2644–2655, 2017.

[109] Y. Su, I. King, and M. R. Lyu. Learning to rank using localized geometric mean metrics. In *SIGIR*, pages 45–54, 2017.

[110] Y. Suh, B. Han, W. Kim, and K. M. Lee. Stochastic class-based hard example mining for deep metric learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7251–7259, 2019.

[111] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.

[112] D. Tingle, Y. E. Kim, and D. Turnbull. Exploring automatic music annotation with "acoustically-objective" tags. In *Proceedings of the international conference on Multimedia information retrieval*, page 55, New York, New York, USA, mar 2010. ACM Press.

[113] J. N. Tsitsiklis, D. P. Bertsekas, M. Athans, et al. Distributed asynchronous deterministic and stochastic gradient optimization algorithms. *IEEE transactions on automatic control*, 31(9):803–812, 1986.

[114] E. Ustinova and V. S. Lempitsky. Learning deep embeddings with histogram loss. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 4170–4178, 2016.

[115] N. Verma, D. Mahajan, S. Sellamanickam, and V. Nair. Learning hierarchical similarity metrics. In *2012 IEEE*

Conference on Computer Vision and Pattern Recognition, Providence, RI, USA, June 16-21, 2012, pages 2280–2287, 2012.

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

[117] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The caltech-ucsd birds-200-2011 dataset. 2011.

[118] J. Wang, A. Kalousis, and A. Woznica. Parametric Local Metric Learning for Nearest Neighbor Classification. In *Advances in Neural Information Processing Systems*, pages 1601–1609, 2012.

[119] J. Wang, L. Yu, W. Zhang, Y. Gong, Y. Xu, B. Wang, P. Zhang, and D. Zhang. IRGAN: A minimax game for unifying generative and discriminative information retrieval models. In *SIGIR*, pages 515–524, 2017.

[120] X. Wang and A. Gupta. Unsupervised learning of visual representations using videos. In *IEEE International Conference on Computer Vision*, pages 2794–2802, 2015.

[121] Y. Wang, V. Sadhanala, W. Dai, W. Neiswanger, S. Sra, and E. P. Xing. Parallel and distributed block-coordinate frank-wolfe algorithms. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, pages 1548–1557, 2016.

[122] P. Watcharapichat, V. L. Morales, R. C. Fernandez, and P. R. Pietzuch. Ako: Decentralised deep learning with

partial gradient exchange. In *Proceedings of the Seventh ACM Symposium on Cloud Computing*, pages 84–97, 2016.

[123] X. Wei, B. Gong, Z. Liu, W. Lu, and L. Wang. Improving the improved training of wasserstein gans: A consistency term and its dual effect. *ICLR*, 2018.

[124] K. Q. Weinberger and L. K. Saul. Distance Metric Learning for Large Margin Nearest Neighbor Classification. *Journal of Machine Learning Research*, 10:207–244, 2009.

[125] J. Weston, S. Bengio, and N. Usunier. Large scale image annotation: Learning to rank with joint word-image embeddings. *Machine Learning*, 81(1):21–35, 2010.

[126] J. Wu, Z. Huang, J. Thoma, D. Acharya, and L. V. Gool. Wasserstein divergence for gans. *ECCV*, 2018.

[127] P. Xie and E. P. Xing. Large scale distributed distance metric learning. *CoRR*, abs/1412.5949, 2014.

[128] E. P. Xing, M. I. Jordan, S. Russell, and A. Y. Ng. Distance metric learning with application to clustering with side-information. In *Advances in neural information processing systems*, pages 505–512, 2002.

[129] C. Xiong, D. Johnson, R. Xu, and J. J. Corso. Random forests for metric learning with implicit pairwise position dependence. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '12, pages 958–966, New York, NY, USA, 2012. ACM.

[130] S. Xiong, Y. Zhang, D. Ji, and Y. Lou. Distance metric learning for aspect phrase grouping. *arXiv preprint arXiv:1604.08672*, 2016.

[131] Z. E. Xu, K. Q. Weinberger, O. Chapelle, S. Louis, and O. C. Cc. The Greedy Miser: Learning under Test-time Budgets. In *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, pages 1175–1182, 2012.

[132] Y. Ying and P. Li. Distance metric learning with eigenvalue optimization. *Journal of Machine Learning Research*, 13:1–26, 2012.

[133] F. Yu, Y. Zhang, S. Song, A. Seff, and J. Xiao. LSUN: construction of a large-scale image dataset using deep learning with humans in the loop. *CoRR*, abs/1506.03365, 2015.

[134] Y. Yuan, K. Yang, and C. Zhang. Hard-aware deeply cascaded embedding. In *International Conference on Computer Vision*, pages 814–823, 2017.

[135] P. H. Zadeh, R. Hosseini, and S. Sra. Geometric mean metric learning. In *Proceedings of the 33rd International Conference on Machine Learning*, 2016.

[136] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica. Spark: cluster computing with working sets. In *Proceedings of the 2nd USENIX conference on Hot topics in cloud computing*, volume 10, page 10, 2010.

[137] H. Zhang, Z. Zheng, S. Xu, W. Dai, Q. Ho, X. Liang, Z. Hu, J. Wei, P. Xie, and E. P. Xing. Poseidon: An

efficient communication architecture for distributed deep learning on GPU clusters. In *2017 USENIX Annual Technical Conference, USENIX ATC 2017, Santa Clara, CA, USA, July 12-14, 2017.*, pages 181–193, 2017.

[138] J. Zhang, X. Shi, S. Zhao, and I. King. Star-gcn: Stacked and reconstructed graph convolutional networks for recommender systems. In *IJCAI*, 2019.

[139] S. Zhao, X. Chen, I. King, and M. R. Lyu. Personalized sequential check-in prediction: Beyond geographical and temporal contexts. In *ICME*, pages 1–6, 2018.

[140] S. Zhao, T. Zhao, H. Yang, M. R. Lyu, and I. King. Stellar: spatial-temporal latent ranking for successive point-of-interest recommendation. In *AAAI*, 2016.

[141] M. Zinkevich, M. Weimer, A. J. Smola, and L. Li. Parallelized stochastic gradient descent. In *Advances in Neural Information Processing Systems*, pages 2595–2603, 2010.