2016

Content-based Information Retrieval via Nearest Neighbor Search

Yinjie Huang
University of Central Florida

CONTENT-BASED INFORMATION RETRIEVAL VIA NEAREST NEIGHBOR SEARCH

by

YINJIE HUANG
B.S. Dalian University of Technology, 2010
M.S. University of Central Florida, 2014

A dissertation submitted in partial fulfilment of the requirements
for the degree of Doctor of Philosophy
in the Department of Electrical Engineering and Computer Science
in the College of Engineering and Computer Science
at the University of Central Florida
Orlando, Florida

Summer Term
2016

Major Professors: Michael Georgiopoulos, Georgios C. Anagnostopoulos

ABSTRACT

Content-based information retrieval (CBIR) has attracted significant interest in the past few years. When given a search query, the search engine will compare the query with all the stored information in the database through nearest neighbor search. Finally, the system will return the most similar items. We contribute to the CBIR research the following: firstly, Distance Metric Learning (DML) is studied to improve retrieval accuracy of nearest neighbor search. Additionally, Hash Function Learning (HFL) is considered to accelerate the retrieval process.

On one hand, a new local metric learning framework is proposed - Reduced-Rank Local Metric Learning (R^2LML). By considering a conical combination of Mahalanobis metrics, the proposed method is able to better capture information like data's similarity and location. A regularization to suppress the noise and avoid over-fitting is also incorporated into the formulation. Based on the different methods to infer the weights for the local metric, we considered two frameworks: Transductive Reduced-Rank Local Metric Learning (T-R^2LML), which utilizes transductive learning, while Efficient Reduced-Rank Local Metric Learning (E-R^2LML) employs a simpler and faster approximated method. Besides, we study the convergence property of the proposed block coordinate descent algorithms for both our frameworks. The extensive experiments show the superiority of our approaches.

On the other hand, *Supervised Hash Learning (*SHL), which could be used in supervised, semi-supervised and unsupervised learning scenarios, was proposed in the dissertation. By considering several codewords which could be learned from the data, the proposed method naturally derives to several Support Vector Machine (SVM) problems. After providing an efficient training algorithm, we also study the theoretical generalization bound of the new hashing framework. In the final experiments, *SHL outperforms many other popular hash function learning methods. Additionally,

in order to cope with large data sets, we also conducted experiments running on big data using a parallel computing software package, namely *LIBSKYLARK*.

This thesis work is dedicated to my parents, Junxiong Huang and Qiuqin Xu, who have always loved me unconditionally and taught me to work hard deligently. This work is also dedicated to my brother Hua Feng, my sister-in-law Jianhong Shen and my nephew Yicheng Feng, for their constant support and encouragement during the challenges of my graduate life.

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my advisors Dr. Michael Georgiopoulos and Dr. Georgios C. Anagnostopoulos for their continuous support of my Ph.D study and related research. Their guidance helped me in all the time of research and writing of this thesis. Moreover, I acknowledge support from UCF Graduate College Trustee Fellowship, CSIT TEAm grant from the Florida Board of Governors and National Science Foundation (NSF) grant No. 1200566 [1] and No. 0963146 [2].

Besides my advisors, I would like to thank the rest of my thesis committee: Dr. Haiyan Nancy Hu, Dr. Gita Reese Sukthankar, and Dr. Liqiang Ni, for their insightful comments and encouragement.

Additionally, I thank my labmates, Cong Li, Tiantian Zhang, Niloofar Yousefi and Mahlagha Sedghi for the discussions and for all the fun we have had in the last few years.

Eventually, I would like to thank my family: my parents and my brother and his family for supporting me throughout writing this thesis.

[1] http://www.nsf.gov/awardsearch/showAward?AWD_ID=1200566

[2] http://www.nsf.gov/awardsearch/showAward?AWD_ID=0963146&HistoricalAwards=false

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: INTRODUCTION

With the explosive growth of the World Wide Web, information has become widely accessible to the world. To efficiently browse and search for the useful knowledge among this large amount of information, a user needs an Information Retrieval (IR) system [107]. Nowadays people engage in IR every day. For instance, web search engines like the ones of Google [1], Yahoo [2] or Bing [3] are IR systems. Even the email client program employs an IR system, which enables the users to search emails or texts by keywords. Additionally, there are plenty of applications related to IR. For example, traders are able to retrieve stock prices and sales information from financial or marketing databases. Doctors can search for similar X-rays, CT or MRI scans to help diagnose diseases. Finally, law enforcement officers are able to search for likely suspects or check fingerprints in a database.

The definition of IR, extracted from [74], is provided below:

Definition 1. *Information Retrieval (IR) is finding material (usually documents) of an unstructured nature (usually texts) that satisfies an information need from within large collections (usually stored on computers).*

Therefore, an IR system is a computer system for retrieving information from a large digital database. All the digital information is indexed and stored in the database. When a search query is performed by a user, the search engine looks up the index to match the given query with the stored information. Eventually, the retrieved results are provided in descending order according to a similarity measure. Additionally, IR also supports users in browsing or filtering the stored

[1] `https://www.google.com/`

[2] `https://search.yahoo.com/`

[3] `http://www.bing.com/`

information or even processing the retrieved information.

IR systems can be distinguished into three categories by the scale, at which they operate. In *web search*, since it is necessary to search over billions of documents stored across millions of computers, an IR system is designed to handle this enormous scale efficiently. Popular search engine websites like *Google* and *Bing* fall into this category. At the other extreme is *personal information retrieval*, which runs on personal computers to handle everyday tasks, such as searching for files, filtering documents etc. For example, Apple's Max OS X Spotlight [4] and Microsoft Windows Search [5] are two typical personal information retrieval systems. This IR system needs to be designed sufficiently lightweight in terms of processing and disk space usage that it can run on a single computer without annoying its user. The third category is the *enterprise/institutional and domain-specific search*, which provides information retrieval for collections, such as ones stored in a corporation's database. Generally, this type of IR system performs the search on a handful of machines.

Given different approaches defining the similarity measure, IR systems can also be divided into two categories - concept-based information retrieval and content-based information retrieval.

Concept-based information retrieval [61], a description-based information retrieval system, retrieves from text-based indexing of information. The text information includes keywords, captions or annotations. However, there are several drawbacks of this IR system. First, the system may highly depend on the keywords, which are the main source of defining the similarity measure, without considering the semantic meaning of the word. Given an IR system building on keyword lists, one could easily find a synonymous word so that the system will retrieve wrong results. Additionally, concept-based information retrieval systems perform poorly, when confronting erroneous

[4] `https://support.apple.com/en-us/HT204014`

[5] `https://msdn.microsoft.com/en-us/library/windows/desktop/aa965362(v=vs.85).aspx`

or incomprehensive captions and annotations [49]. For example, when the captions or annotations contain noise, the system may retrieve wrong results based on the query given by the user.

On the other hand, content-based information retrieval [58] analyzes the contents of the information resources. Instead of text labels, keywords or annotations, this system utilizes various low-level features to index information resources. Content-based methods are necessary when text annotations, labels and keywords are nonexistent, incomplete or noisy. Furthermore, this method can potentially improve retrieval performance and give additional insight about the media collections by being combining with text information. Thus, since 2000, content-based information retrieval literature has grown tremendously. Researchers and experts from various research areas like machine learning, computer vision, human-computer interfaces and information retrieval are contributing actively to the content-based information retrieval community.

One family of content-based system analyzes text content to generate features to eventually retrieve the information. The system utilizes natural language processing techniques to analyze syntax and semantics of the text to categorize the document. For example, Bags of Words (BoW) can be used in cross-lingual information retrieval [41]. Other examples of such a systems are provided in [115].

Most content-based systems are designed to retrieve images. [108] clearly illustrated and summarized at a high level about the procedures of image retrieval. Much research work in image retrieval tries to extract the visual content from images. The earliest work of image indexing relates to color histograms [111]. [30] considers light shape and reflection. They also studied color consistency which tries to detect the same color when environmental color changes. [43] also proposed color correlograms to enhance histograms. Gabor filters were also used in local shape extraction for matching and retrieving of images, as shown in [73]. Additionally, there are many real world applications in content-based image retrieval systems. For public use, there are Google Image Search and Yahoo Image Search. For other uses, content-based IR systems are widely used

for family album management, astronomy and remote sensing [140] [126] [22] [87] [99]. Finally, interested readers can find comprehensive surveys about content-based image retrieval in [1] [95] [108] and [109].

Since data are, in most cases, stored and represented as numerical vectors, many content-based IR systems employ k-nearest neighbors (KNN) search [72] [114] [32]. This approach sorts retrieved items in terms of their distances/similarities to the query item. In most cases, the system utilizes the Euclidean distance metric. Finally, fixed k-nearest items are retrieved by the system as an answer to the user's query. However, there are several drawbacks of this KNN search algorithm:

- The choice of Euclidean distance metric may not be appropriate for all information retrieval tasks. Some features with large numeric range may dominate the distance value. Therefore, an approach that automatically weights features is necessary.
- The computation of KNN search is expensive, especially when the system copes with a large amount of data. What's more, the storage of big data also poses a problem. A new method is necessary to accelerate the retrieval process and reduce the storage cost.

This dissertation contributes to the content-based IR field in the following perspectives:

- To overcome the drawbacks of the simple usage of Euclidean distance metric, Distance Metric Learning (DML) is studied.
- A new local DML model is proposed to improve the retrieval accuracy using KNN.
- The new DML method considers low rank regularization, which could suppress noise and handle sparse data.

- Two training algorithms, whose convergence analysis have been studied, are provided for two scenarios of our DML formulations.
- In order to accelerate the retrieval process, Hash Function Learning (HFL) is utilized. The binary codes from the learned hash functions represent the original data.
- A new HFL is proposed, which successfully incorporates supervised, semi-supervised and unsupervised hash function learning.
- We also show the theoretical generalization bound for our HFL framework.

The rest of the dissertation is organized as follows:

- CHAPTER 2 introduces our motivation for content-based information retrieval. The motivation contains two aspects:
  - DML is utilized to improve KNN search accuracy.
  - To accelerate the retrieval process, HFL is studied.
- CHAPTER 3 describes DML and proposes our method - Reduced-Rank Local Metric Learning (R^2LML). In particular, there are two versions of R^2LML: one utilizes transductive learning, namely Transductive Reduced-Rank Local Metric Learning (T-R^2LML); the other one is a simpler and faster version, namely Efficient Reduced-Rank Local Metric Learning (E-R^2LML). Two algorithms are provided for each of the frameworks along with their convergence analysis. The experiments show that, when compared with other state-of-art DML frameworks, both E-R^2LML and T-R^2LML show impressive results.
- CHAPTER 4 introduces HFL, which is used to accelerate the retrieval process in IR systems. Recent HFL methods can be grouped into three learning scenarios: supervised, unsupervised

and semi-supervised learning. We propose a method, *Supervised Hash Learning (*SHL), which can address all three learning scenarios in a single framework. Moreover, our framework naturally derives to several Support Vector Machine (SVM) problems, which could be trained efficiently by many off-the-shelf solvers that are on-line available. An theoretical generalization analysis of *SHL's superior performance is also studied. After experimenting with our framework and other popular HFL methods on several benchmarks, we conclude that *SHL yields very competitive and, occasionally impressive results.

- CHAPTER 5 concludes this dissertation. Additionally, there are also some future work or potential research areas based on the work discussed in this chapter. For instance, the idea of hashing could be extended in multi-label learning problems. Moreover, the proposed metric learning framework could also be used in many applications.

- All the detailed proofs for various propositions, lemmas or theorems can be found in the APPENDIX.

CHAPTER 2: MOTIVATION

Performance improvement via Distance Metric Learning (DML)

Similarity and distance judgments play a very important role in some human cognitive processes, such as recognition and categorization. As human beings, we tend to provide similar responses or take similar actions, when facing stimuli similar to what we have encountered before. Based on this, many psychologists have developed many cognitive theories and mathematical models of similarity in [2] [39] [75] and [77].

Given its studies, it is no surprise that the concept of similarity and distance computation play important roles in many machine learning, pattern recognition and data mining methods. For classification, k-nearest neighbors (KNN) [21] utilizes a metric distance to identify the nearest neighbors. For clustering, k-means [71] relies on distance measurement. For an Information Retrieval (IR) system [74], results are often ranked according to similarity-based relevance to a given query.

How to appropriately measure the similarity or distance between objects is crucial to the performance of all the above methods. When handling numerical data, the Euclidean metric can be used in the model. However, a general-purpose distance metric (e.g., the Euclidean distance) may not be appropriate for all data sets and may lead to suboptimal performance. Psychological studies also support this hypothesis. For example, depending on the context, [36] and [85] indicate that humans weight features differently. To illustrate this point, suppose we have a database of images. If we plan to find matching faces based on identity, the similarity should be based on appropriate features like hair color, face shape, etc. If we have another application to determine the pose of an individual, which may require the other features in this scenario. Therefore, in real world application, even for a domain expert, it is very difficult to define the optimal similarity measure for a

specific task.

In order to automatically learn similarity measures from data and improve performance, DML is proposed. Since DML's first appearance as a convex optimization probem in [131], it has become an active research community with many papers published and numerous DML algorithms proposed at highly regarded computer vision, data mining and machine learning venues.

Generally, the goal of DML is to adapt some pairwise metric function, for instance, the Mahalanobis distance $d_A(\boldsymbol{x}, \boldsymbol{x}') = \sqrt{(\boldsymbol{x} - \boldsymbol{x}')^T A(\boldsymbol{x} - \boldsymbol{x}')}$, to the problem of interest. Most methods learn the metric (here, the positive semi-definite matrix A in d_A) in a supervised way from pair of instances. The pair of instances are categorized as similar and dissimilar sets of the following forms:

$$\mathcal{S} = \{(\boldsymbol{x}_i, \boldsymbol{x}_j) : \boldsymbol{x}_i \text{ and } \boldsymbol{x}_j \text{ are similar}\}$$
$$\mathcal{D} = \{(\boldsymbol{x}_i, \boldsymbol{x}_j) : \boldsymbol{x}_i \text{ and } \boldsymbol{x}_j \text{ are dissimilar}\}$$

In order to learn a metric from a data set, a DML algorithm is basically formulated as an optimization problem, which has the following general form:

$$\min_A \; l(A) + \lambda R(A) \tag{2.1}$$

where l is a loss function and R is the regularizer on the parameters A. $\lambda \geq 0$ is the regularization/penalty parameter. Different choices of loss function and regularizations will lead to various

DML algorithms.

The previous general form only defines one global metric matrix for all the distance computations in the data set. However, one single, global metric for all distance computations may not be well-suited for all problem settings. Therefore, in this dissertation, to improve nearest neighbor search accuracy for IR system, a local distance metric learning framework is proposed. Here, local information includes similarity characteristics and location of the data considered. The new framework shows the following characteristics:

- The framework considers the local metric as a conical combination of Mahalanobis metrics. Both the coefficients of the combination and the Mahalonobis weight matrices are learned from data.
- When considering the nuclear norm on metric matrix, the proposed formulation successfully controls the rank of Mahalanobis matrix.
- Two formulations are introduced: one involves transductive learning, while the other formulation is a simpler and faster version.
- In order to solve the two formulations, we provide two Block Coordinate Descent (BCD) algorithms respectively. Due to the closed form solution in some block minimization, the proposed algorithms are very efficient.
- Theoretical results about the proposed algorithms' convergence analysis is also showcased in the dissertation. We show that the proposed algorithms contain Karush-Kuhn-Tucker (KKT) points.
- In the experiments, not only we show the relationship between the number of local metrics and the accuracies of the model, we also compare our framework with the other popular DML models and achieve impressive results.

Acceleration of retrieval process via Hash Function Learning (HFL)

With the thriving development of the World Wide Web (WWW), all the web-related data such as documents, images and videos are growing explosively. Nowadays, the WWW contains over 366 millions websites, which include more than one trillion webpages [1]. For instance, the eBook subscription service, Scribd [2] hosts about 60 million documents on its platform. Social network platform Twitter [3] receives more than 100 million tweets per day. Flicker [4], the photo sharing website, has over 5 billion images. The images are still being uploaded at a rate of over 3000 per minute in Flicker. Yahoo! [5] exchanges over 3 billion messages every day. The video sharing website YouTube [6] receives more than 100 hours of uploaded videos per minute from different users.

Modern information technology infrastructure needs to be designed to handle this huge amount of data. Compared to the storage bottleneck, searching efficiently for relevant content in these gigantic databases becomes an even more challenging task; searching for media data like audio, videos and images remains a challenge in practical applications. In the past few years, content-based information retrieval has attracted lots of interests to help alleviate this situation. Basically, instead of just relying on textual keywords or annotations, content-based system tries to index media content with the extracted features from the information, like color and shape from images or frequency from audios.

Generally, content-based IR relates to nearest neighbor search [101] - a popular method in machine

[1] This number is estimated by Google Web Index 2008.

[2] https://www.scribd.com/

[3] https://www.twitter.com

[4] https://www.flickr.com/

[5] https://www.yahoo.com/

[6] https://www.youtube.com/

learning. However, it is impractical if the system tries to compare the query with all the samples in the database, when handling huge databases nowadays. Instead, we need a search algorithm faster than nearest neighbor search's linear complexity $\mathcal{O}(|\mathcal{X}|)$, where $\mathcal{X}$ is the given database. Besides, the *curse of dimensionality* [6] still poses a huge problem in most content-based IR applications, since visual features normally contain hundreds or thousands of dimensions. Finally, storage also becomes a critical bottleneck, since the system has to load all the original data into memory.

One acceleration technique is called Approximate Nearest Neighbors (ANN) search. The idea of this method is, instead of returning an accurate nearest neighbor search, one can provide an approximate result with sublinear search complexity. For instance, given query q, ϵ-ANN [48] tries to find $p \in \mathcal{X}$ satisfying $d(p, q) \leq (1 + \epsilon)d(p', q)$, where $\epsilon > 0$ and p' is the exact nearest neighbor of given query q.

One type of ANN methods is tree-based and has been popular during the past several decades. Many tree-based algorithms, which are able to achieve $\mathcal{O}(log(|\mathcal{X}|))$ complexity, are proposed, such as ball tree [86], KD tree [8], vantage point tree [133] and metric tree [119]. However, tree-based methods still require huge storage for data. Additionally, when handling high-dimensional data, the performance of these tree-based ANN methods deteriorates.

On the other hand, hashing-based ANN approaches have attracted much attention in the recent few years. By mapping the input data to a discrete Hamming code, the approaches not only can achieve efficient retrieval speed, but they can also substantially reduce storage cost, since the system only needs to store the compact binary codes instead of the original data. Suppose we transform 80 million tiny images [117], which are 32×32 pixels, into 64-bit compact binary codes. If they are stored as double-floating points, one is able to reduce the storage from 600 GB to 600 MB.

Hashing approaches have been intensively studied in many different research areas like computer graphics, computational geometry, computer vision and telecommunications for many years. Re-

cently, many hashing methods have been developed to utilize machine learning techniques to generate more effective binary codes. The goal of learning-to-hash is to learn data-dependent hash functions to generate hash codes to achieve good search or retrieval accuracy. A hashing method is defined as follows: suppose we have N data samples $\mathbf{X} \in \mathbb{R}^{D \times N}$, the purpose of learning-to-hashing is to learn a B-bit binary codes $\mathbf{Y} \in \mathbb{R}^{B \times N}$. To generate these binary codes, B hash functions are designed. One example is the linear hash function, which is widely used in various recent works [33], [91]. In linear hash function learning, the bth bit can be generated as the following form:

$$h_b(\boldsymbol{x}) = \mathrm{sgn}(f(\boldsymbol{w}_b^T \boldsymbol{x} + d_b)) \tag{2.2}$$

where $\boldsymbol{x}$ is a data point, $\boldsymbol{w}_b$ and d_b are hash function parameters. Different choices of these parameters lead to different learned hash functions. A good hashing approach should generate the binary codes that preserve the similarity consistency, i.e., similar images should have similar codes.

In this work, a novel hash function learning approach - *Supervised Hash Learning (*SHL) is proposed, which contains the following merits:

- *SHL utilizes several Hamming codewords to group data. When learned during training, these Hamming codewords represent various classes in the Hamming space.
- The proposed model can incorporate all three learning scenarios since both unlabeled and labeled data can contribute in the learning process.
- By adopting Multiple Kernel Learning (MKL) approach to infer a suitable kernel, we show that the formulation of *SHL naturally derives to several Support Vector Machine (SVM)

problems, which can be efficiently solved by some online software, namely *LIBSVM*.

- To optimize the framework, a three-step Block Coordinate Descent algorithm based on an efficient Majorization Minimization (MM) approach is introduced in this work.

- Theoretical analysis of *SHL's superior performance is also provided based on Rademacher complexity.

- Experiments show that the proposed approach could achieve very impressive results compared to the other methods in content-based image retrieval applications. Additionally, experiments on transductive learning and foreground/background image segmentation also show the superiority of *SHL.

CHAPTER 3: METRIC LEARNING

Introduction

Distance computations underlie many machine learning approaches. For example, the KNN classification and the k-Means clustering algorithm are popular in machine learning and data mining. Such computations are often, if not mainly, performed using the ordinary Euclidean metric or a weighted one, namely the Mahalanobis distance. However, employing fixed, global metrics for computing distances, such as the ones just mentioned, may not yield good results in all settings. This fact motivated many researchers to develop data-driven frameworks, which could offer the best metric for a specific problem (*e.g.* [131] and [103]). In successfully addressing this task, one needs to take into account the data's distributional characteristics and to take advantage of any *side information* that may be available for the data. Generally, these methods are named as *Distance Metric Learning*. An example of such a framework is to learn the weight matrix of the Mahalanobis metric, which, often, we will refer to it simply as the metric. Equivalently, this task could be viewed as follows: a linear transformation is learned in the original space and Euclidean distances are computed in the feature space. Eventually, when coping with a classification problem, a KNN search rule based on the learned metric is employed to classify data points.

Our work tries to learn the Mahalanobis metric for a classification problem with the help of pairwise sample similarities. By definition of similarities, if two samples are from the same class, they will be labeled as similar. Metric learning aims to map similar samples close and to map dissimilar samples far apart in the feature space. After learning this metric, a KNN classification exhibits improved accuracy over a direct application of the same search rule using the Euclidean metric.

Many metric learning algorithms have been proposed to show significant improvements over the

Euclidean KNN rule. [131] is the first Mahalanobis distance metric learning framework. Based on two sets of similar or dissimilar pairs, the authors tried to solve a clustering problem. The proposed framework is formulated as a convex problem, trying to minimize the distances between similar samples and maximize the distances between dissimilar samples. In their work, a projected gradient ascent algorithm is employed. At each iteration, a gradient ascent step is taken, followed by a projection onto the positive semidefinite cone, which is done by setting the negative eigenvalues of the metric matrix to zero. Finally, experiments on several data sets show that the proposed framework improves k-Means clustering performance. Some recent work like [14] and [135] revisited this framework by forming the same problem as an eigenvalue optimization, which can be solved faster by their algorithm.

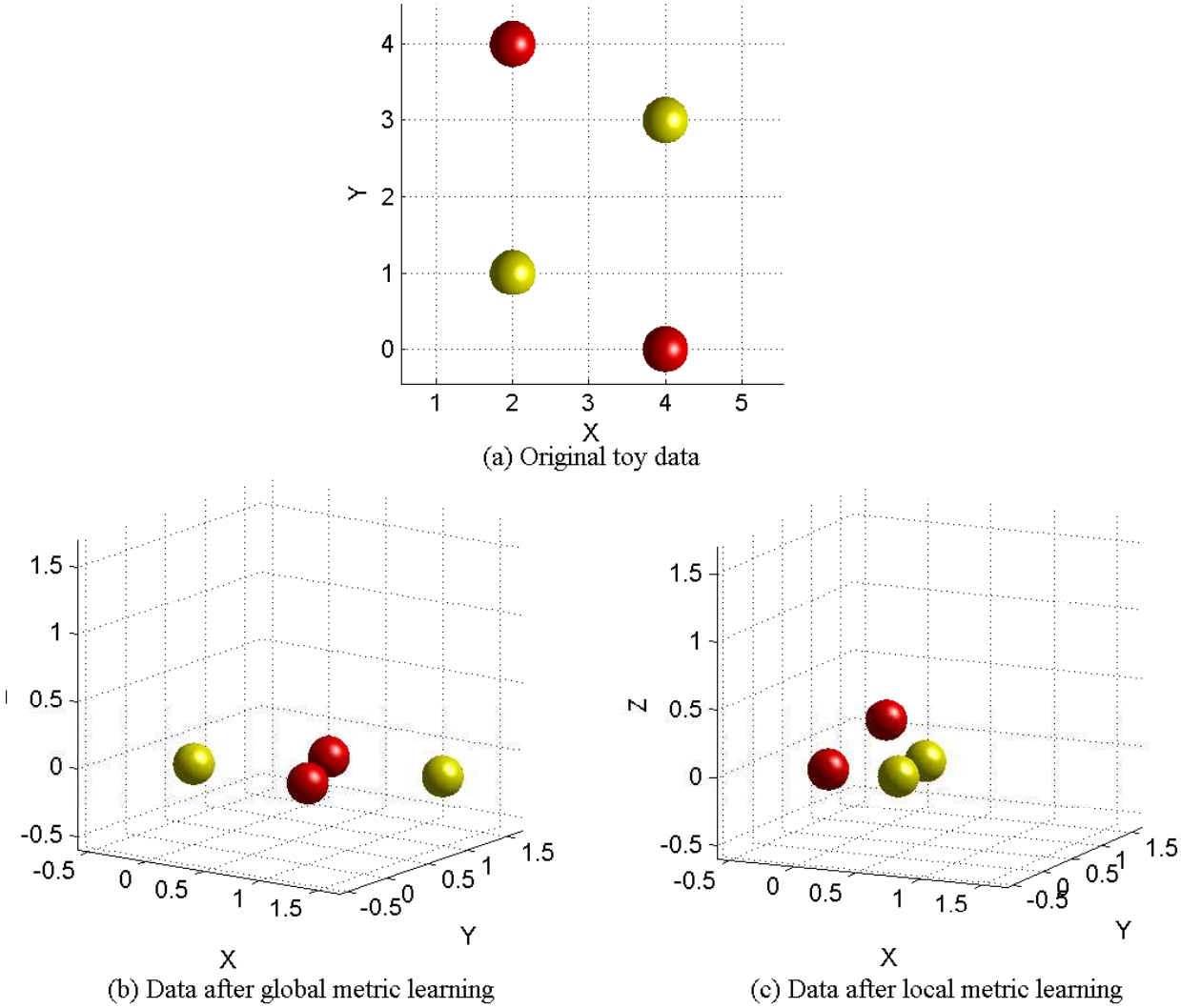
The authors of [103] introduced the first on-line Mahalanobis distance learning framework, which learns both the metric and a threshold b. At each step, the framework will receive a pair of points and the algorithm will perform the projections. [103] also provides an error bound for this framework. The method was further improved by [50], which utilizes *LogDet* divergence regularization and provides tighter error bound. They also provide a more efficient algorithm to optimize the framework. Moreover, their experiments show that the framework offers impressive performance.

In [35], the authors introduced Neighborhood Component Analysis (NCA), which optimizes the leave-one-out error of a stochastic nearest neighbor classifier. Since NCA's formulation is non-convex, gradient descent may get stuck in local minima. However, the experiments show that NCA outperforms the Euclidean based KNN. Note that NCA can also be used to reduce dimensionality of the input data sets. Some additional related recent works include [34] and [113]. The former work, based on KL divergence, proposed an alternative formulation of NCA. This new framework is convex, however, it requires expensive optimization. The latter, instead, generalized NCA to KNN with $k > 1$.

In [20], a system is put forward that maps images to points in a lower dimensional space so that these points lie closer, if the original images are similar. This model consists of two convolutional neural networks to address geometric distortions. This framework was used in face recognition. The experiments on *Purdue* and *AR* face databases show impressive results, although the face images from these databases have a high degree of variability in the lighting, position or expressions.

Large Margin Nearest Neighbor (LMNN) [127] is the most widely known metric learning framework. This algorithm constructs the constraints in the following rules: for every training sample, the k nearest neighbors of the same class should lie closer than the samples of the other classes. Here, the k nearest neighbors is determined by Euclidean distance. After defining the constraints based on these k nearest neighbors, an optimization problem is formulated to pull target neighbors closer, which is optimized by a subgradient descent algorithm. LMNN achieves impressive practical performance compared to the other DML frameworks. Additionally, [27] also pointed out that LMNN can be derived from Support Vector Machines. Finally, one can find many papers working on algorithms to optimize LMNN framework faster like in [82], [90] and [118].

The authors of [134] proposed Sparse Metric Learning (SML), which utilizes a mixed $L_{2,1}$ norm to regularize the metric matrix. This norm tends to set the entire column as zero and performs feature selection. Finally, the authors reformulated SML as a min-max problem and solved it using the algorithm proposed in [81]. Because of SML's feature selection, its performance is slightly better than LMNN in some data sets.



(a) Original toy data

(b) Data after global metric learning

(c) Data after local metric learning

Figure 3.1: Toy data set to exhibit the potential advantages of local metric learning. (a) Original data. (b) After learning a global metic, data in the feature space. (c) After learning several local metrics, data in the feature space.

Information-Theoretic Metric Learning (ITML), proposed in [24], utilizes the *LogDet* regularization for metric learning. The framework tries to regularize the metric matrix to be as close as possible to a prior matrix. For example, this prior matrix can be the identity or the inverse covariance matrix. This problem is optimized by repeatedly performing Bregman projections onto a single constraint. Experiments show that ITML could achieve very impressive results. However, one drawback of ITML is that the choice of the prior matrix may vary in different problem settings; a wrong choice may lead to poor performance.

The aforementioned metric learning approaches share one common feature: they employ a global metric, which is used to compute all distances. However, this global metric learning framework may not be well-suited to multi-modal distributions or non-linear scenarios. Figure 3.1 illustrates this point via a toy data set containing 4 samples from two classes. Note that this toy problem is merely a conceptual device that shows the comparison of what a global metric and local metrics will do in a non-linear scenarios. Figure 3.1(a) illustrates the samples in the original space. After learning a global metric, Figure 3.1(b) shows data samples in the feature space. Figure 3.1(c) shows the feature space after learning two local metrics, which take into considerations the similarity and location of the data. Such metrics are referred as *local metrics*. In contrast to the result obtained using a global metric, local metrics can map similar samples closer, as shown in Figure 3.1(c). Thus, local metric learning may potentially improve 1-NN classification performance.

Many local metric learning algorithms have been proposed. Firstly, [40] introduced a locally adaptive form of KNN to alleviate the curse of dimensionality. In their method, neighborhoods are defined by a metric estimated from local linear discriminant analysis. By defining local metrics using centroid information, the neighbors, in directions parallel to the boundaries, are elongated, while in directions that are orthogonal to the boundaries, are shrank. Their framework could also be used to reduce the dimensionality of the data sets. Finally, the experiments show a substantial improvement of their framework over the KNN decision rule.

Derived from k-Means, [9] proposed MPCK-Means which combines both metric learning and the use of pairwise constraints. MPCK-means utilizes both unlabeled data and similar or dissimilar pairwise constraints and performs distance metric learning for each cluster. Since the approach defines individual metric for each cluster, MPCK-Means generates clusters of different shapes. Experimental results on semi-supervised data sets show that their approach outperforms the other cluster methods.

A local metric learning model introduced in [132] brings pairs from the same mode of a class closer and separates nearby pairs from different classes. In order to solve the framework, the authors propose a probabilistic formulation and an EM-like algorithm is used to solve it. In specific, the algorithm involves eigenvector analysis and bound optimization. Their approach on image classification and text categorization shows to be quite competitive.

LMNN-Multiple Metric (LMNN-MM) [128] is an extension of LMNN to learn several Mahalanobis metrics from the data set. LMNN-MM requires a preprocessing step to partition the data into several clusters. The method defines a local metric for each cluster individually. The problem remains convex and LMNN-MM generates significant improvements over LMNN. However, this framework requires expensive computational cost and it tends to overfit the training data set.

The authors of [83] proposed Generative Local Metric Learning (GLML), which relates to reduce the bias of the nearest neighbor classifier by learning local metrics. By suppressing this bias at each point, the classification error is significantly reduced compared to the global metric learning approaches. The authors also show that optimization on the problem results in a semi-definite programming optimization, whose solutions will generate locally optimal metrics. Eventually, GLML has a strong assumption that the data points are sampled from a Gaussian mixture.

Parametric Local Metric Learning (PLML) [123] tries to learn a Mahalanobis metric for each training instance. The metric is linearly combined with a few matrices which are associated with

an anchor point. PLML deals with the over-fitting problem by considering a Frobenius norm and manifold regularization. However, since PLML requires eigenvalue decompositions at each step, high-dimensional data sets are intractable for PLML. Additionally, PLML considers many hyper-parameters, although the authors fix most of them to empirical values. Finally, the experiments show that PLML outperforms LMNN-MM on many data sets considered.

In [105], the authors introduced Sparse Compositional Metric Learning (SCML), which learns Mahalanobis metrics as sparse combinations of a set of bases. These bases consist of rank-one matrices that are locally discriminative. Their framework utilizes $L_{2,1}$ norms to achieve sparsity and stochastic gradient is employed to optimize this non-convex framework. In practice, it is shown that SCML outperforms LMNN-MM in terms of classification accuracies.

Multi-granularity neighborhood distance metric learning (MGML) introduced in [143] learns multiple distance metrics under different scales of granularity. Here, granularity refers to the level of a hierarchy of information. The authors first evaluate a distance metric based on neighborhood margin. The approach is then formulated as a SVM model, which can be efficiently solved by many SVM solvers. Finally, the various decisions from different granularity are combined with the learned weights to compute the results. The experiments show that the proposed approach is really competitive, when considering many popular data sets.

In this work, a new local metric learning framework is proposed, namely Reduced-Rank Local Metric Learning (R^2LML). As elaborated in subsection **Problem Formulation**, in our approach, the local Mahalanobis metric (in specific, its weight matrix) is modeled as a conical combination of positive semi-definite weight matrices. With the assistance of pair-wise similarities, both the weight matrices and their coefficients are learned from the data. The weight matrices themselves correspond to local linear transformations of the original data from their native space into a locality-dependent feature space. These transformations are learned such that similar (dissimilar) samples

map close to (far from) each other, so that they exhibit small (large) pair-wise Euclidean distances in these locally-defined feature spaces. Note that, in our case, we will consider samples to be similar, if they share the same label. Moreover, we will consider two variants of R^2LML. The first one, namely Transductive Reduced-Rank Local Metric Learning (T-R^2LML), uses transductive learning [120] to infer the test sample coefficients necessary for defining the local metrics. The second one, which is referred to as Efficient Reduced-Rank Local Metric Learning (E-R^2LML), aims to address the computationally intensive nature of the first variant. As discussed in subsection **Problem Formulation**, it employs a technique first used in [123]. The coefficients of a test sample are set equal to the ones of its nearest training sample. Here, nearest samples are defined using Euclidean distances. Finally, it is worth mentioning that both variants employ a sum-of-nuclear-norms regularizer to avoid over-fitting, when warranted.

In order to optimize the aforementioned formulations, two efficient BCD algorithms are presented in subsection **Algorithm**. In specific, as delineated in subsection **Algorithm** for E-R^2LML, a two-block minimization algorithm is able to solve the E-R^2LML learning problem. The first block minimization with respect to the weight matrices constitutes a Proximal Subgradient Descent (PSD) step, which is able to cope with the non-smooth nature of the formulation's regularizer. The second block minimization, which attempts to optimize the metric coefficients, constitutes a straightforward MM step. On the other hand, the algorithm intended for solving the T-R^2LML formulation differs from the first one in that it includes an additional block minimization with respect to the test samples' similarities. As shown in subsection **Algorithm**, for T-R^2LML, the relevant optimization, while addressing a binary integer programming problem, can be efficiently performed. The convergence analysis for both methods is showcased in subsection **Analysis**.

Finally, in subsection **Experiments**, the first experiment studies the importance of regularization in the proposed frameworks based on the synthetic data sets. Additionally, the relationship between the number of local metrics and the accuracies is highlighted in the second experiment. Eventually,

we demonstrate the capabilities of T-R^{2}LML and E-R^{2}LML w.r.t classification problems. After compared with other popular global or local metric learning frameworks, T-R^{2}LML and E-R^{2}LML achieve the highest classification accuracy in 9 and 14 out of 18 data sets respectively.

Problem Formulation

For any positive integer M, define $\mathbb{N}_M \triangleq \{1, 2, \ldots, M\}$. Suppose there are n input training set $\{\boldsymbol{x}_n \in \mathbb{R}^D\}_{n \in \mathbb{N}_N}$ and an similarity matrix $\boldsymbol{S} \in \{0, 1\}^{N \times N}$. In this matrix, which is served as side information, each entry represents a corresponding pair-wise sample similarity. If $\boldsymbol{x}_m$ and $\boldsymbol{x}_n$ are dissimilar, then $s_{mn} = 0$; otherwise, then $s_{mn} = 1$. In a classification context, two samples from the same class can be defined as similar.

The Mahalanobis distance between two samples is $d_{\boldsymbol{A}}(\boldsymbol{x}_m, \boldsymbol{x}_n) \triangleq \sqrt{(\boldsymbol{x}_m - \boldsymbol{x}_n)^T \boldsymbol{A} (\boldsymbol{x}_m - \boldsymbol{x}_n)}$. The positive semi-definite (PSD) matrix, $\boldsymbol{A} \in \mathbb{R}^{D \times D}$, will be referred as the *weight matrix of the metric*. When $\boldsymbol{A} = \boldsymbol{I}$, the previous metric becomes the Euclidean metric. Since any PSD weight matrix can be decomposed as $\boldsymbol{A} = \boldsymbol{L}^T \boldsymbol{L}$, where $\boldsymbol{L} \in \mathbb{R}^{P \times D}$ with $P \leq D$, the Mahalanobis distance can be reformulated as $d_{\boldsymbol{A}}(\boldsymbol{x}_m, \boldsymbol{x}_n) = \|\boldsymbol{L}(\boldsymbol{x}_m - \boldsymbol{x}_n)\|_2$. This expression indicates that the Mahalanobis distance based on $\boldsymbol{A}$ between two points in the original space, can be viewed as the Euclidean distance between the points transformed by $\boldsymbol{L}$ in the feature space.

Metric learning aims to learn $\boldsymbol{A}$ so that the distances between pairs of similar points are minimized. Meanwhile, the distances between dissimilar points should maintain above a certain threshold. We can formulate the problem as follows:

$$\begin{aligned} \underset{\boldsymbol{A}\succeq 0}{min} \ & \sum_{m,n} s_{mn} d_{\boldsymbol{A}}(\boldsymbol{x}_m, \boldsymbol{x}_n) \\ s.t. \ & \sum_{m,n} (1 - s_{mn}) d_{\boldsymbol{A}}(\boldsymbol{x}_m, \boldsymbol{x}_n) \geq 1. \end{aligned} \tag{3.1}$$

Based on $\boldsymbol{A}$, Problem (3.1) is a semi-definite programming problem. Several approaches like LMNN, ITML and NCA are learning a single global metric. However, as argued earlier via Figure 3.1, a global metric may not offer best performances under any circumstance.

In this dissertation, R^{2}LML, a new local metric approach, is proposed. We assume that the metric in our problem is expressed as a conical combination of $K \geq 1$ Mahalanobis metrics. The metric between $\boldsymbol{x}_n$ and $\boldsymbol{x}_m$ is defined as $d_{MA}(\boldsymbol{x}_m, \boldsymbol{x}_m) \triangleq \sum_k \boldsymbol{A}^k g_m^k g_n^k$. The vector $\boldsymbol{g}^k \in \mathbb{R}^N$ is defined for each local metric k, of which the nth element g_n^k may be considered as a measure of how pertinent the kth metric is, when calculating distances considering the nth sample. Not only do these metrics change throughout the input space along the data's underlying manifold, but are also affected by the similarity of nearby samples. Note that these coefficient vectors will be also unknown for test samples and, hence, need to be inferred as well. A natural avenue to achieve this is via a transductive learning scheme.

The metric $\sum_k \boldsymbol{A}^k g_m^k g_n^k$ is actually defined as a semi-metric [100], since it violates the triangle inequality; for some choices of $\boldsymbol{g}$, there exist triplets of samples that do not satisfy the triangle inequality in the feature space. However, in our experiments, it seems that a proper metric is almost always learned. For example, when considering the *Pendigits* data set (containing about 200 samples), the triangle inequalities that we examined (over one million) were all satisfied. In the rest of our work, we still refer to this semi-metric as a metric for simplicity.

Transductive learning trains both labeled and unlabeled data to yield improved performance. According to [120], when solving a problem, one should avoid inferring a function as an intermediate step. There are many transductive learning approaches proposed for various algorithms. In [7], [18], [31] and [52], the authors developed a transductive learning framework for Support Vector Machine. [51] and [54] introduced a transductive algorithm for KNN classifiers and general classifiers respectively. There are also transductive learning approaches for graph-based models in [112], [65] and [142].

In T-R^{2}LML, the input training set $\{\boldsymbol{x}_n \in \mathbb{R}^D\}_{n\in\mathbb{N}_N}$ and test set $\{\boldsymbol{x}_n \in \mathbb{R}^D\}_{n\in\mathbb{N}_M}$ are combined. The entries of the similarity matrix $\boldsymbol{S} \in \{0,1\}^{(N+M)\times(N+M)}$ that involve test data are randomly initialized. The vectors $\boldsymbol{g}^k \in \Omega_g' \triangleq \left\{\{\boldsymbol{g}_k\}_{k\in\mathbb{N}_K} \in [0,1]^{N+M} : \boldsymbol{g}^k \succeq \mathbf{0},\ \sum_k \boldsymbol{g}^k = \mathbf{1}\right\}$. Note that '$\succeq$' is component-wise ordering. The constraint that the $\boldsymbol{g}^k$s' need to sum up to the all-ones vector $\mathbf{1}$ means that, when computing distances from each sample, one metric is relevant at least. Apparently, when $K = 1$, $\boldsymbol{g}^1 = \mathbf{1}$, which leads to learn a single global metric.

The weight matrix for each pair (m, n) can be defined as $\sum_k \boldsymbol{A}^k g_m^k g_n^k$ based on the previous description. Note that, for every pair of points, when computing distances, a different weight matrix is employed. We now consider the following formulation motivated by Problem (3.1), which varies over $k \in \mathbb{K}$:

$$
\begin{aligned}
\underset{\boldsymbol{L}^k,\boldsymbol{S},\boldsymbol{g}^k\in\Omega_g',\xi_{m,n}^k\geq 0}{min} \quad & \sum_k\sum_{m,n} s_{mn}\left\|\boldsymbol{L}^k\Delta\boldsymbol{x}_{mn}\right\|_2^2 g_n^k g_m^k + \\
& + C\sum_k\sum_{m,n}(1-s_{mn})\xi_{mn}^k + \lambda\sum_k \text{rank}(\boldsymbol{L}^k) \\
s.t. \quad & \left\|\boldsymbol{L}^k\Delta\boldsymbol{x}_{mn}\right\|_2^2 \geq 1-\xi_{mn}^k, \quad m,n\in\mathbb{N}_{N+M},\ k\in\mathbb{N}_K \\
& s_{mn}\in\{0,1\},\ m,n\in\mathbb{N}_M \\
& s_{mm}=1,\ s_{mn}=s_{nm},\ m,n\in\mathbb{N}_M \\
& \sum_{n\in\mathbb{N}_{N+M}} s_{mn}\geq 2,\ m\in\mathbb{N}_M,
\end{aligned}
\tag{3.2}
$$

where $\Delta\boldsymbol{x}_{mn} \triangleq \boldsymbol{x}_m - \boldsymbol{x}_n$ and the matrix $\boldsymbol{L}^k$'s rank is denoted as rank($\boldsymbol{L}^k$). In the objective function, the distance between similar samples is minimized by the first term. For pairs of dissimilar samples, with the slack variables ξ_{mn}^k, the second term encourages distances to be larger than 1. Here, $C > 0$ controls the penalty of violating the previous prerequisite. Eventually, the last term penalizes large ranks of $\boldsymbol{L}^k$. Thus, the dimensionality of the feature space is actually controlled by $\lambda \geq 0$, the regularization parameter. As is typical for identifying good values for regularization parameters, both C and λ are chosen via a validation procedure. Also, note that the diagonal elements are all set to 1 in the similarity matrix. Finally, the last constraint guarantees that the testing samples include all the labels of the training set, since one test sample at least shares the same label as the training data.

Via the use of the hinge function, $[u]_+ \triangleq \max\{u, 0\}$ for all $u \in \mathbb{R}$, by eliminating the slack variables, Problem (3.2) could be reformulated. Notice that rank($\boldsymbol{L}^k$) is difficult to be directly optimize over, since it is a non-convex function with respect to $\boldsymbol{L}^k$. As illustrated in [13] and [12], we can replace rank($\boldsymbol{L}^k$) with its convex envelope, which is actually $\boldsymbol{L}^k$'s nuclear norm. The new

problem is now formulated as:

$$
\begin{aligned}
\underset{\boldsymbol{L}^k,\boldsymbol{S},\boldsymbol{g}^k\in\Omega_g'}{min} \quad & \sum_k\sum_{m,n} s_{mn}\left\|\boldsymbol{L}^k\Delta\boldsymbol{x}_{mn}\right\|_2^2 g_n^k g_m^k + \\
& + C(1-s_{mn})\left[1-\left\|\boldsymbol{L}^k\Delta\boldsymbol{x}_{mn}\right\|_2^2\right]_+ + \lambda\sum_k\left\|\boldsymbol{L}^k\right\|_* \\
s.t. \quad & s_{mn}\in\{0,1\},\ m,n\in\mathbb{N}_M \\
& s_{mm}=1,\ s_{mn}=s_{nm},\ m,n\in\mathbb{N}_M \\
& \sum_{n\in\mathbb{N}_{N+M}} s_{mn}\geq 2,\ m\in\mathbb{N}_M,
\end{aligned}
\tag{3.3}
$$

where $\|\cdot\|_*$ denotes the nuclear norm, in specific, $\left\|\boldsymbol{L}^k\right\|_* \triangleq \sum_{s=1}^{P}\sigma_s(\boldsymbol{L}^k)$, where σ_s is a singular value of $\boldsymbol{L}^k$.

A shortcoming of T-R^{2}LML is that, it is computationally intensive, since the computation of the gradient in each step requires $O(K(M+N)^2)$ operations and, typically, $M >> N$. Hence, we are also inclined to consider a faster, albeit approximate, approach to address our local metric learning problem. In specific, as done in [123] and [45], for each test sample $\boldsymbol{x}$, its $\boldsymbol{g}$ vector will be assigned the value of $\boldsymbol{x}$'s nearest training sample's $\boldsymbol{g}$. Here, nearest samples are searched using Euclidean distances. We refer to this model as E-R^{2}LML and its training only requires $O(KN^2)$ operations per step.

For E-R^{2}LML, $\boldsymbol{g}^k \in \Omega_g \triangleq \left\{\{\boldsymbol{g}_k\}_{k\in\mathbb{N}_K}\in[0,1]^N : \boldsymbol{g}^k\succeq\boldsymbol{0},\ \sum_k\boldsymbol{g}^k=\boldsymbol{1}\right\}$ when considering only the training set. Finally, the problem becomes:

$$\min_{\boldsymbol{L}^k, \boldsymbol{g}^k \in \Omega_g} \sum_k \sum_{m,n} s_{mn} \left\| \boldsymbol{L}^k \Delta \boldsymbol{x}_{mn} \right\|_2^2 g_n^k g_m^k + \\ + C(1 - s_{mn}) \left[1 - \left\| \boldsymbol{L}^k \Delta \boldsymbol{x}_{mn} \right\|_2^2 \right]_+ + \lambda \sum_k \left\| \boldsymbol{L}^k \right\|_* . \tag{3.4}$$

All the other parameters like λ and C in Problem (3.4) are defined as in Problem (3.3).

Algorithm

Problem (3.4) and Problem (3.3) reflect minimizations over two and three sets of variables respectively. In E-R^{2}LML, for fixed $\boldsymbol{g}^k$, with respect to $\boldsymbol{L}^k$, the problem is non-convex, since the second term in Eq. (3.4) combines a non-monotone function w.r.t $\boldsymbol{L}^k$, namely $1 - \left\| \boldsymbol{L}^k \Delta \boldsymbol{x}_{mn} \right\|_2^2$ and a convex function (hinge function). Moreover, for fixed $\boldsymbol{L}^k$, the problem is also non-convex w.r.t $\boldsymbol{g}^k$, since the similarity matrix $\boldsymbol{S}$ is indefinite, which will be showcased in the following section. Thus, there may exist several minima in the objective function. Therefore, we may need to initialize an iterative algorithm many times with different values in order to find a good solution. The same observations apply to T-R^{2}LML as well. Finally, notice that, for T-R^{2}LML, when optimizing Problem (3.3) w.r.t $\boldsymbol{S}$, while holding $\boldsymbol{g}^k$ and $\boldsymbol{L}^k$ fixed, the problem under consideration is convex. In what follows next, we discuss two training algorithms: a two-block BCD algorithm for E-R^{2}LML and a very similar BCD algorithm for T-R^{2}LML that can perform the optimizations in question.

Algorithm for E-R2LML

We first start off with a discussion of the BCD that trains the E-R^{2}LML framework. For the first block, we try to solve for every $\boldsymbol{L}^k$ by holding the $\boldsymbol{g}^k$'s fixed. In this case, Problem (3.4) becomes a

minimization problem without constraints, which can be expressed in the form $f(\boldsymbol{w})+r(\boldsymbol{w})$. Here $\boldsymbol{w}$ is the parameter we are trying to optimize over (in our case, all $\boldsymbol{L}^k$'s). The non-differentiable hinge loss function is $f(\boldsymbol{w})$, while $r(\boldsymbol{w})$ is a non-smooth, convex regularization term. Hence, we employ a Proximal Subgradient Descent method as in [93] and [17]. It might be worth noting that the particular approach is a special case of the one presented in [28]. It is this relationship that we leverage to develop the convergence analysis of our PSD steps in subsection **Analysis**.

Next, in the second block, we minimize with respect to each $\boldsymbol{g}^k$ vector, while the $\boldsymbol{L}^k$'s are assumed to be fixed. Consider a matrix $\bar{\boldsymbol{S}}^k$ associated to the kth metric. The (m,n) element in this matrix is defined as:

$$\bar{s}_{mn}^k \triangleq s_{mn} \left\| \boldsymbol{L}^k \Delta \boldsymbol{x}_{mn} \right\|_2^2, \quad m,n \in \mathbb{N}_N. \tag{3.5}$$

Then, if concatenating all individual $\boldsymbol{g}^k$ vectors into a single vector $\boldsymbol{g} \in \mathbb{R}^{KN}$ and by defining the block-diagonal matrix $\tilde{\boldsymbol{S}}$ as:

$$\tilde{\boldsymbol{S}} \triangleq \begin{bmatrix} \bar{\boldsymbol{S}}^1 & 0 & ... & 0 \\ 0 & \bar{\boldsymbol{S}}^2 & ... & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & ... & 0 & \bar{\boldsymbol{S}}^K \end{bmatrix} \in \mathbb{R}^{KN \times KN}. \tag{3.6}$$

Problem (3.4) can be expressed as:

$$\min_{\boldsymbol{g}\in\boldsymbol{\Omega}_{\boldsymbol{g}}} \ \boldsymbol{g}^T\tilde{\boldsymbol{S}}\boldsymbol{g}, \tag{3.7}$$

Problem (3.7) needs to satisfy the constraint $\boldsymbol{B}\boldsymbol{g} = \mathbf{1}$, where $\boldsymbol{B} \triangleq \mathbf{1}^T \otimes \boldsymbol{I}_N$ and $\otimes$ denotes the Kronecker product. Since $\tilde{\boldsymbol{S}}$ is indefinite, Problem (3.7) is non-convex. This is because $\tilde{\boldsymbol{S}}$ is a block diagonal matrix with Euclidean Distance Matrices (EDMs) as its blocks. EDMs contain only one positive eigenvalue (unless all of them equal to 0). Since EDM matrix is hollow, its trace is 0. Thus, the remaining eigenvalues of EDM matrix must be negative, as illustrated in [3]. Therefore, $\tilde{\boldsymbol{S}}$ contains negative eigenvalues.

In order to minimize Problem (3.7), a Majorization Minimization approach [47] is utilized. To do so, a function of $\boldsymbol{g}$, which majorizes the objective function, needs to be defined. Let $\mu \triangleq -\lambda_{max}(\tilde{\boldsymbol{S}})$, where $\lambda_{max}(\tilde{\boldsymbol{S}})$ is the largest eigenvalue of $\tilde{\boldsymbol{S}}$. Since $\tilde{\boldsymbol{S}}$ is indefinite, $\lambda_{max}(\tilde{\boldsymbol{S}}) > 0$. Thus, $\boldsymbol{H} \triangleq \tilde{\boldsymbol{S}} + \mu\boldsymbol{I}$ will be negative semi-definite. Suppose $q(\boldsymbol{g}) \triangleq \boldsymbol{g}^T\tilde{\boldsymbol{S}}\boldsymbol{g}$ is defined as the cost function of Eq. (3.7). Note that $(\boldsymbol{g}-\boldsymbol{g}')^T\boldsymbol{H}(\boldsymbol{g}-\boldsymbol{g}') \leq 0$ for any $\boldsymbol{g}$ and $\boldsymbol{g}'$ and we have that $q(\boldsymbol{g}) < -\boldsymbol{g}'^T\boldsymbol{H}\boldsymbol{g}' + 2\boldsymbol{g}'^T\boldsymbol{H}\boldsymbol{g} - \mu\left\|\boldsymbol{g}\right\|_2^2$ for all $\boldsymbol{g} \neq \boldsymbol{g}'$ and equality, only if $\boldsymbol{g} = \boldsymbol{g}'$. The right hand side of the aforementioned inequality constitutes q's majorizing function, denoted as $q(\boldsymbol{g}|\boldsymbol{g}')$. $\boldsymbol{g}$ is iteratively optimized by majorizing function based on the current estimate $\boldsymbol{g}'$. Therefore, the convex problem w.r.t $\boldsymbol{g}$ is presented as follows:

$$\min_{\boldsymbol{g}\in\Omega_g} \ 2\boldsymbol{g}'^T\boldsymbol{H}\boldsymbol{g} - \mu\left\|\boldsymbol{g}\right\|_2^2. \tag{3.8}$$

As the next theorem implies, Problem (3.8) can be readily solved.

Theorem 1. *Let $\boldsymbol{g}, \boldsymbol{d} \in \mathbb{R}^{KN}$, $\boldsymbol{B} \triangleq \mathbf{1}^T \otimes \boldsymbol{I}_N \in \mathbb{R}^{N \times KN}$ and $c > 0$. The unique minimizer $\boldsymbol{g}^*$ of*

$$\begin{aligned} \min_{\boldsymbol{g}} \ & \frac{c}{2} \|\boldsymbol{g}\|_2^2 + \boldsymbol{d}^T \boldsymbol{g} \\ s.t. \ & \boldsymbol{B}\boldsymbol{g} = \mathbf{1}, \ \boldsymbol{g} \succeq \mathbf{0}, \end{aligned} \tag{3.9}$$

has the form

$$g_i^* = \frac{1}{c} \left[(\boldsymbol{B}^T \boldsymbol{\alpha})_i - \boldsymbol{d}_i \right]_+, \quad i \in \mathbb{N}_{KN}, \tag{3.10}$$

where g_i is the ith element of $\boldsymbol{g}$ and $\boldsymbol{\alpha} \in \mathbb{R}^N$ is the Lagrange multiplier vector associated to the equality constraint.

Proof. For Problem (3.9), the Lagrangian is:

$$L(\boldsymbol{g}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \frac{c}{2} \boldsymbol{g}^T \boldsymbol{g} + \boldsymbol{d}^T \boldsymbol{g} + \boldsymbol{\alpha}^T (\mathbf{1} - \boldsymbol{B}\boldsymbol{g}) - \boldsymbol{\beta}^T \boldsymbol{g}, \tag{3.11}$$

where both $\boldsymbol{\alpha} \in \mathbb{R}^N$ and $\boldsymbol{\beta} \in \mathbb{R}^{KN}$ are Lagrange multiplier vectors. Note that we have $\boldsymbol{\beta} \succeq \mathbf{0}$. If we set the partial derivative of $L(\boldsymbol{g}, \boldsymbol{\alpha}, \boldsymbol{\beta})$ w.r.t $\boldsymbol{g}$ to $\mathbf{0}$, we have

$$g_i = \frac{1}{c} \left((\boldsymbol{B}^T \boldsymbol{\alpha})_i + \beta_i - d_i \right), \quad i \in \mathbb{N}_{KN}. \tag{3.12}$$

Let $\gamma_i \triangleq (\boldsymbol{B}^T \boldsymbol{\alpha})_i - d_i$. If we combine the complementary slackness condition $\beta_i g_i = 0$ with Eq. (3.12), one obtains that, if $\gamma_i \leq 0$, then $\beta_i = -\gamma_i$ and $g_i = 0$, while, when $\gamma_i > 0$, then $\beta_i = 0$

and, evidently, $g_i = \frac{1}{c}\gamma_i$. These two observations can be summarized as $g_i = \frac{1}{c}\left[\gamma_i\right]_+$. $\square$

From Theorem 1, in order to obtain a concrete solution to Problem (3.8), we utilize a binary search to find the optimal values of α_i, so that the equality constraint $\boldsymbol{B}\boldsymbol{g} = \mathbf{1}$ is satisfied.

In conclusion, as depicted Algorithm 1, to solve Problem (3.4), the entire algorithm can be described as follows: for the first block, when the $\boldsymbol{g}^k$ are fixed, we utilize a PSD algorithm to minimize Eq. (3.4) with respect to each matrix $\boldsymbol{L}^k$. In the second block, all $\boldsymbol{L}^k$'s are held fixed. The solution provided in Theorem 1 along with binary search solutions for the α_i's are utilized to compute the optimal $\boldsymbol{g}_k$'s by iteratively solving Problem (3.8) via a Majorization Minimization algorithm. These two blocks will be repeated until the convergence is established.

Algorithm for T-R2LML

The first two BCD steps of T-R^{2}LML are identical to the ones of E-R^{2}LML. However, since T-R^{2}LML embodies a trasductive learning approach, a third BCD step is required, in order to predict the similarities between all samples, including the ones used for testing. In specific, for the third block optimization, Problem (3.3) is minimized over $\boldsymbol{S}$ for fixed $\boldsymbol{L}^k$'s and $\boldsymbol{g}^k$'s. By defining

$$\psi_{mn} \triangleq \sum_k \left(\left\| \boldsymbol{L}^k \Delta \boldsymbol{x}_{mn} \right\|_2^2 g_n^k g_m^k - C[1 - \left\| \boldsymbol{L}^k \Delta \boldsymbol{x}_{mn} \right\|_2^2]_+ \right), \tag{3.13}$$

Problem (3.3) becomes:

$$
\begin{aligned}
\min_{\boldsymbol{S}} \;\; & \text{trace}\,\{\boldsymbol{S}\boldsymbol{\Psi}\} \\
s.t. \;\; & s_{mn} \in \{0,1\},\; m,n \in \mathbb{N}_M \\
& s_{mm} = 1,\; s_{mn} = s_{nm},\; m,n \in \mathbb{N}_M \\
& \sum_{n \in \mathbb{N}_{N+M}} s_{mn} \geq 2,\; m \in \mathbb{N}_M.
\end{aligned} \tag{3.14}
$$

where $\boldsymbol{\Psi} \in \mathbb{R}^{M \times N}$ is the matrix with elements ψ_{mn}. This is a $0-1$ integer programming problem. By scanning the matrix $\boldsymbol{\Psi}$ row by row, Problem (3.14) will be optimally solved using the following rules:

- For rows of $\boldsymbol{\Psi}$ containing at least one negative element, set the corresponding s_{mn} element(s) to 1; the remaining elements are set to 0.
- For rows of $\boldsymbol{\Psi}$ with no negative element, the s_{mn} element, which corresponds to the smallest ψ_{mn}, is set to 1; the remaining elements are set to 0.
- Note that s_{nm} must equal s_{mn}, since the matrix $\boldsymbol{S}$ is symmetric.

For the sake of completeness, in Algorithm 1, we summarize the relevant algorithm. Note that these three main blocks are repeated until a preset maximum number of steps is reached.

Analysis

In this subsection, the convergence analysis of Algorithm 1 is investigated. This is a local analysis, since our framework is non-convex. As mentioned in previous sections, the function $f(\boldsymbol{w}) + r(\boldsymbol{w})$

Algorithm 1 Minimization of Problem (3.3) and Problem (3.4)

Input: Data $\boldsymbol{X} \in R^{D\times(N+M)}$ for Problem (3.3) and $\boldsymbol{X} \in R^{D\times N}$ for Problem (3.4), number of metrics K

Output: $\boldsymbol{L}^k, \boldsymbol{g}^k, \boldsymbol{S}$ (here, $\boldsymbol{S}$ is only for Problem (3.3))

01. Initialize $\boldsymbol{L}^k, \boldsymbol{g}^k, \boldsymbol{S}$ for all $k \in \mathbb{N}_K$
02. **While** not converged **Do**
03. Block 1: For each $\boldsymbol{L}^k$, a PSD method to solve Problem (3.3) is employed
04. Block 2:
05. $\tilde{\boldsymbol{S}} \leftarrow Eq.\ (3.6)$
06. $\mu \leftarrow -\lambda_{max}(\tilde{\boldsymbol{S}})$
07. $\boldsymbol{H} \leftarrow \tilde{\boldsymbol{S}} + \mu \boldsymbol{I}$
08. **While** not converged **Do**
09. Using Eq. (3.10), binary search is applied to obtain each $\boldsymbol{g}^k$
10. **End While**
11. Block 3 (this block only for Problem (3.3)): Optimal Algorithm for Problem (3.14)
11. **End While**

is minimized using a Proximal Subgradient Descent approach, where both $r \triangleq \sum_k \left\|\boldsymbol{L}^k\right\|_*$ and $f \triangleq \sum_k \sum_{m,n} s_{mn} \left\|\boldsymbol{L}^k \Delta \boldsymbol{x}_{mn}\right\|_2^2 g_n^k g_m^k + C(1-s_{mn}) \left[1 - \left\|\boldsymbol{L}^k \Delta \boldsymbol{x}_{mn}\right\|_2^2\right]_+$ are non-differentiable. ∂f is defined as the subgradient of f and let's define $\|\partial f(\boldsymbol{w})\| \triangleq \sup_{\boldsymbol{g}\in\partial f(\boldsymbol{w})} \|\boldsymbol{g}\|_2$; Like in [57] and [104], the subgradients are assumed to be bounded:

$$\|\partial f(\boldsymbol{w})\|^2 \leq Af(\boldsymbol{w}) + G^2, \quad \|\partial r(\boldsymbol{w})\|^2 \leq Ar(\boldsymbol{w}) + G^2, \tag{3.15}$$

where A and G are positive scalars. Let $\boldsymbol{w}^*$ be the minimizer of $f(\boldsymbol{w}) + r(\boldsymbol{w})$. Then we have the following theorem for the problem under consideration.

Theorem 2. *Suppose that a PSD method is employed to solve* $\min_{\boldsymbol{w}}\{f(\boldsymbol{w}) + r(\boldsymbol{w})\}$. *Assume that 1) f and r are lower-bounded; 2) the norms of any subgradients ∂f and ∂r are bounded as in Eq. (3.15); 3) $\|\boldsymbol{w}^*\| \leq D$ for some $D > 0$; 4) $r(\mathbf{0}) = 0$. Let $\eta_t \triangleq \frac{D}{\sqrt{8TG}}$, where T is the number of iterations of the PSD algorithm. Then, for a constant $c \leq 4$, such that $(1 - cA\frac{D}{\sqrt{8TD}}) > 0$, and initial estimate of the solution $\boldsymbol{w}_1 = \mathbf{0}$, we have:*

$$\min_{t\in\mathbb{N}_T}[f(\boldsymbol{w}_t)+r(\boldsymbol{w}_t)] \leq \frac{1}{T}\sum_{t=1}^{T} f(\boldsymbol{w}_t)+r(\boldsymbol{w}_t) \leq$$
$$\leq \frac{2\sqrt{2}DG}{\sqrt{T}(1-\frac{cAD}{G\sqrt{8T}})} + \frac{f(\boldsymbol{w}^*)+r(\boldsymbol{w}^*)}{1-\frac{cAD}{G\sqrt{8T}}}. \tag{3.16}$$

The detailed proof of Theorem 2 is given in APPENDIX A. Theorem 2 indicates that the PSD iterates approach $\boldsymbol{w}^*$ as T grows, .

Theorem 3. *Algorithm 1 yields a convergent, non-increasing sequence of cost function values relevant to Problem (3.4) and Problem (3.3). Furthermore, the set of fixed points of the iterative map embodied by Algorithm 1 include the KKT points of Problem (3.4) and Problem (3.3).*

The proof is showcased in APPENDIX B. Theorem 3 implies the convergence of the two proposed algorithms.

Experiments

Effects of our nuclear norm-based regularization

Since T-R^{2}LML involves $(M-1)M/2+MN+(D^2+M+N)K$ parameters, while E-R^{2}LML employs $(D^2+N)K$, both frameworks may benefit from regularization, when confronted with scarce, high-dimensional, noisy data. Two synthetic data sets were created to study nuclear norm.

The first set consisted of 30-dimensional samples, while the second one consisted of 60-dimensional features. In both cases, samples were drawn from a mixture of two highly overlapping Gaussian distributions, whose covariance matrices had a spectral radius of 0.3. Moreover, in the second data

set, features randomly selected with probability 0.5 were set to 0 to emulate sparsity. For both data sets, 80 samples were used for training via E-R^{2}LML and 320 samples for testing. Also, 3 local metrics were employed, while the remaining parameters were set as follows: the step length of PSD was set to 10^{-6} and the algorithm was allowed to run for 5 epochs of 500 iterations each. The classification accuracy using a 5-nearest neighbor search is reported in Table 3.1 and Table 3.2 for various values of the regularization parameters.

The two aforementioned tables reflect, as expected, that the regularization proves to be very important for E-R^{2}LML, and, by extension, to T-R^{2}LML as well, since the latter one deals with additional parameters to be learned. More specifically, it is shown that cross-validation over λ is essential in improving classification accuracy for noisy, potentially sparse, highly overlapping data. This is especially more pronounced for the second data set, where not employing regularization is clearly inferior to the performance attained by fine-tuning λ. Also, for the same data set, Table 3.2 illustrates the sparsity-inducing properties of the nuclear norm regularizer. It is worth noting that, although not specifically shown here, the metrics' all-0 columns obtained for $\lambda = 10^3$ and $\lambda = 10^4$ followed exactly the sparsity pattern of the relevant features.

Real data sets

In order to assess the utility of the proposed models, extensive experiments on 18 data sets were performed, namely, *Robot Navigation*, *Wine Quality*, *Ionosphere*, *Letter Recognition*, *Gamma Telescope*, *Pendigits*, *Breast Tissue*, *Glass*, *Heart*, *Sonar*, *WPBC*, *Optdigits* and *Isolet* data sets from the *University of California, Irvine (UCI) machine learning repository*[1], and *Image Segmentation*, *Two Norm*, *Ring Norm* data sets from the *Delve Dataset Collection*[2]. We also considered

[1] `http://archive.ics.uci.edu/ml/datasets.html`

[2] `http://www.cs.toronto.edu/~delve/data/datasets.html`

Table 3.1: Classification accuracy versus λ for highly overlapping data set.

λ	0	0.1	1	10	100	1000
ACCURACY	0.544	0.528	0.553	0.563	0.563	0.534

Table 3.2: Classification accuracy versus λ for highly overlapping data set with sparse features. The number of columns with all 0 entries for each metric is also reported.

λ	0	0.1	1	10	10^2	10^3	10^4	10^5
ACCURACY	0.725	0.813	0.747	0.713	0.725	0.975	0.916	0.488
# OF ZERO COLUMNS IN METRIC 1	0	0	0	0	0	13	13	0
# OF ZERO COLUMNS IN METRIC 2	0	0	0	0	0	13	13	60
# OF ZERO COLUMNS IN METRIC 3	0	0	0	0	0	13	14	60

the *Columbia University Image Library (COIL20)*[3] and *USPS*[4] data sets. Table 3.3 summarizes the major characteristics of these data sets. Following experimental settings similar to the ones used in [123] and [143], PCA was used on the data of *COIL20*, *Isolet*, *Optdigits* and *USPS* to reduce their number of features to 30, as shown in Table 3.3.

We first explored how the performance of T-R^{2}LML[5] and E-R^{2}LML[6] varies w.r.t the number of local metrics. After this, T-R^{2}LML and E-R^{2}LML are compared to other state-of-the-art global and local metric learning algorithms, namely, ITML [24], LMNN [127], LMNN-MM [128], GLML [83] and PLML [122]. These methods are widely used as baseline approaches in the other metric learning works since they achieve impressive results.

[3] `http://www.cs.columbia.edu/CAVE/software/softlib/coil-20.php`

[4] `http://www.gaussianprocess.org/gpml/data/`

[5] `https://github.com/yinjiehuang/R2LMTL/archive/master.zip`

[6] `https://github.com/yinjiehuang/R2LML/archive/master.zip`

Table 3.3: Benchmark data sets. The columns indicate number of features (#D), classes (#classes), number of validation (#validation) and test (#test) samples.

	#D	#CLASSES	#TRAIN	#VALIDATION	#TEST
A. ROBOT	4	4	240	240	4976
B. LETTER	16	26	520	2600	2600
C. PENDIGITS	16	10	400	2000	2000
D. WINE QUALITY	12	2	150	150	2898
E. TELESCOPE	10	2	300	300	5400
F. IMAGE SEGMENTATION	18	7	210	210	1890
G. TWO NORM	20	2	250	250	3900
H. RING NORM	20	2	250	250	3900
I. IONOSPHERE	34	2	80	50	221
J. BREAST TISSUE	9	6	18	18	70
K. COIL20	30	20	400	400	640
L. GLASS	9	6	18	18	178
M. HEART	13	2	40	40	190
N. ISOLET	30	26	520	3640	3637
O. OPTDIGITS	32	10	400	2400	2820
P. SONAR	60	2	40	40	180
Q. USPS	30	10	400	4500	4398
R. WPBC	33	2	20	20	158

Number of local metrics for T-R2LML and E-R2LML

One aspect that was investigated is how the performances of T-R^{2}LML and E-R^{2}LML vary when different number of local metrics K is considered. The authors of [128] fix K equal to the number of classes for all data sets. This heuristic choice might not necessarily be the optimal. An abundance of data may imply that more local metrics may be necessary for improved performance; this is an aspect we examined for T-R^{2}LML and E-R^{2}LML. For all data sets, the range of K we considered was $1-7$, which, aside from *COIL20*, *USPS* and *Isolet*, included the number of classes represented in the data. As we will argue in the sequel, the optimal K does not coincide with the number of classes of the corresponding classification problem. Actually, it coincides only in

roughly one quarter of the cases.

For T-R^{2}LML, the regularization parameter λ is set to 10 and the penalty parameter C to 1. In the case of E-R^{2}LML, λ was chosen smaller, since it employs less parameters compared to T-R^{2}LML and, therefore, is less prone to over-fitting. Note that all aforementioned parameter values were selected via cross-validation and subsequently held fixed. Moreover, the algorithm will be terminated in two cases: firstly, it reaches 5 epochs; secondly, between two consecutive iterations, the difference of cost function values is less than 10^{-4}. In each epoch, we ran 500 iterations of PSD with step length 10^{-5} for the *Sonar* data set, 10^{-6} for the *Ionosphere* and *Glass* data sets, 10^{-8} for the *Ring Norm* data set, 10^{-9} for the *Robert*, *Letter*, *Two Norm* and *Heart* data sets, 10^{-10} for the *COIL20*, *Isolet*, *Optdigits* and *USPS* data sets, 10^{-11} for the *Pendigits*, *Image Segmentation*, *Telescope*, *Wine Quality* and *Wpbc* data sets and 10^{-13} for the *Breast Tissue* data set. The algorithm terminates the MM loop in two cases: firstly, the algorithm reaches 3000 iterations; secondly, between two iterations, the difference in cost function was less than 10^{-3}.

For E-R^{2}LML, the parameters like C, the number of epochs and the number of iterations were set the same as T-R^{2}LML. The step length of PSD was set to 10^{-3} for the *Glass* and *Sonar* data sets, to 10^{-5} for the *Ionosphere* and *Robot* data sets, to 10^{-6} for the *Two Norm*, *Letter*, *Optdigits* and *Ring Norm* data sets, to 10^{-7} for the *Isolet* and *USPS* data sets, to 10^{-8} for the *Heart*, *Image Segmentations*, *COIL20* and *Wine Quality* data sets, to 10^{-9} for the *Pendigits*, *Gamma Telescope* and *Wpbc* data sets and to 10^{-11} for the *Breast Tissue* data set.

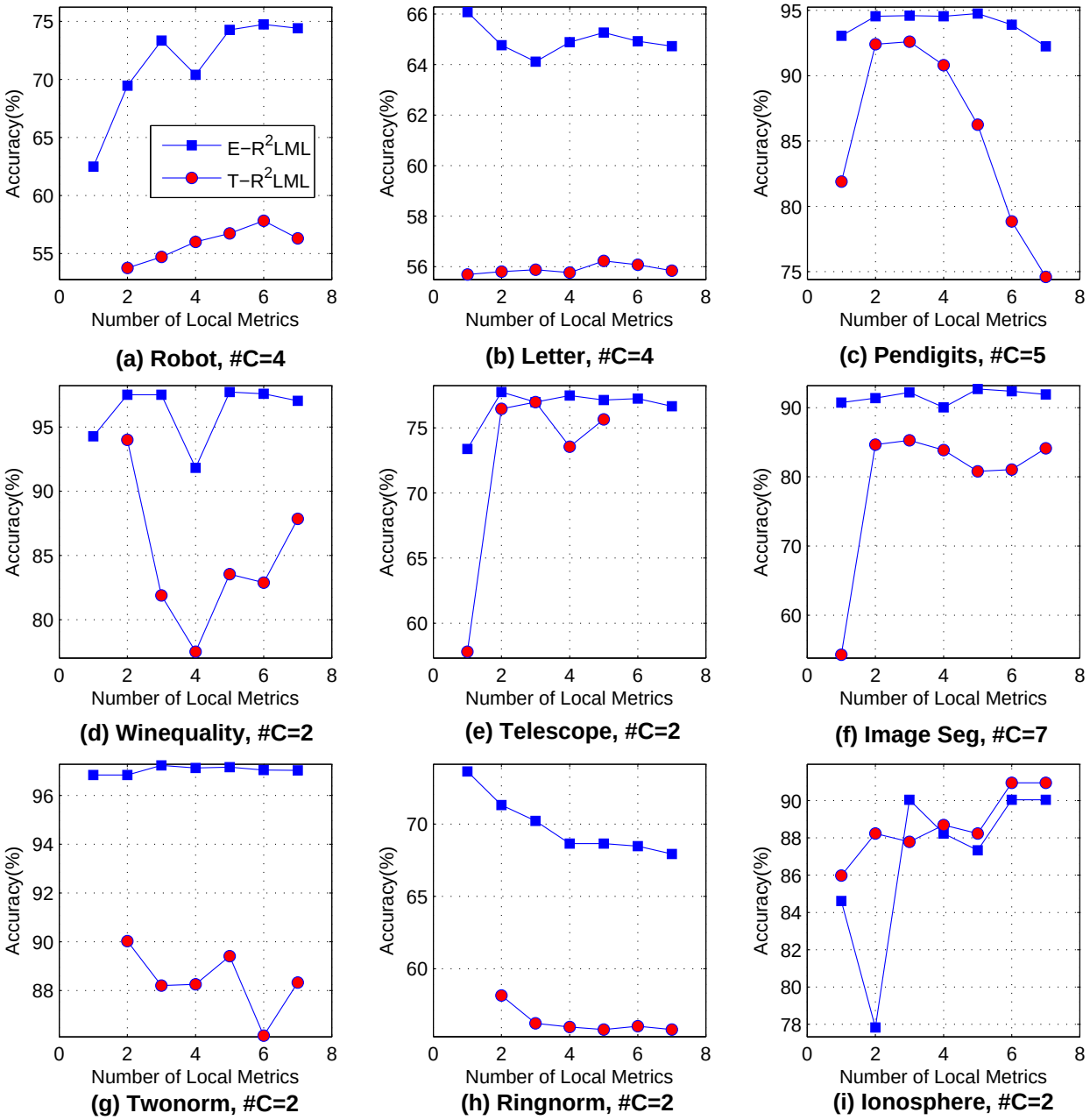


Figure 3.2: When varying number of local metrics, T-R^{2}LML and E-R^{2}LML classification performances on the first 9 benchmark data sets. Here, C denotes the class number.

The classification results when setting different number of local metrics for each data set is reported in Figure 3.2 and Figure 3.3. Several observations can be made based on these results. First, the results indicate that training with more data does not necessarily imply that an increased value of K is needed for improved performance results. For example, in the case of T-R^{2}LML, for the *Pendigits*, *Wine Qulity*, *Two Norm*, *Ring Norm*, *Glass*, *Isolet* and *Optdigits* data sets, 2 local metrics are enough to yield the best results among other choices of K. Additionally, when E-R^{2}LML is trained with the *Telescope* and *USPS* data sets, superior results are obtained using only 2 metrics. Secondly, between the number of local metrics and the accuracy, one can not observe deterministic relationship. The classification accuracy of *Ring Norm* data set is monotonically decreasing w.r.t K. However, with respect to their number of classes for the other data sets, the optimal K changes in a non-obvious manner. All these observations recommend that, in order to achieve the best performing model, validation over K is necessary. Also, one discerns that, although T-R^{2}LML is trained with more data, E-R^{2}LML outperforms it on all data sets except the *Telescope*, *Ionosphere*, *Breast Tissue*, *Heart* and *Wpbc* data sets. T-R^{2}LML does not show much improvement compared to E-R^{2}LML, which will also be illustrated in the next section. Finally, from the obtained results results, it becomes apparent that, using both R^{2}LML variants as local metric learning methods (when $K > 1$) is, more often than not, advantageous compared to the case, when they are used with a global metric; this is most prominently exhibited in the case of the *Heart*, *Wpbc*, *Ionosphere* and *Telescope* data sets.

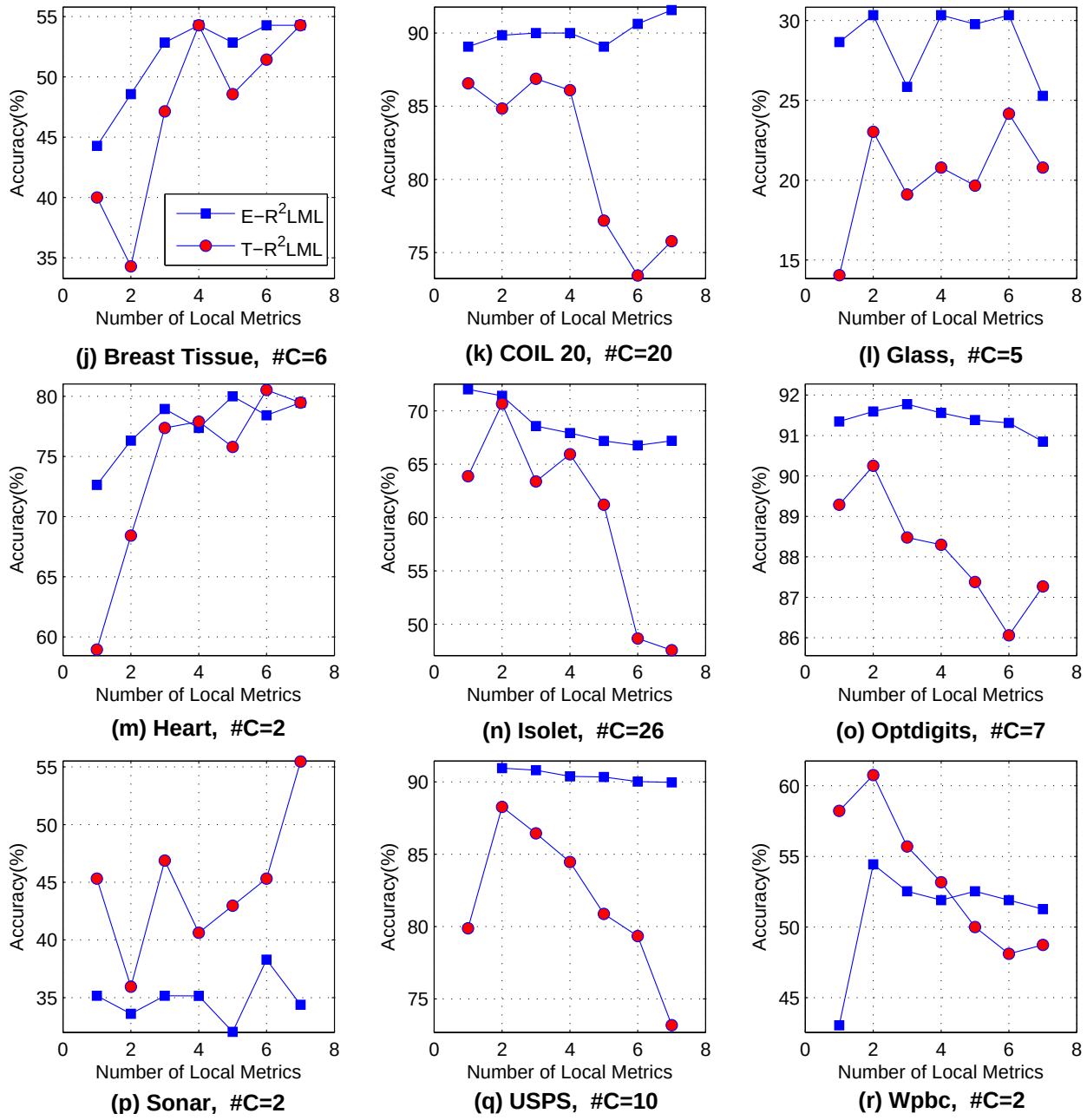


Figure 3.3: When varying number of local metrics, T-R^{2}LML and E-R^{2}LML classification performances on the remaining 9 benchmark data sets. Here, C denotes the class number.

Performance Comparisons

We compared T-R^{2}LML and E-R^{2}LML to several popular state-of-art metric learning algorithms, including LMNN [127], ITML [24], GLML [83], PLML [123] and LMNN-MM [128]. We also consider Euclidean metric KNN as baseline method. As introduced, LMNN and ITML are both global metric learning frameworks, while for GLML, LMNN-MM and PLML, several local metrics are learned. After the metrics are learned for each method, a 5-nearest neighbor decision rule was employed to classify unlabeled samples.

In our experiments, LMNN, LMNN-MM[7], ITML[8] and PLML[9] implementations that are online accessible are used. To be specific, for ITML, we cross-validated over γ to find a good value. Also, the number of attracting neighbors during training for LMNN and LMNN-MM was fixed to 1, as suggested by the authors in the paper. Moreover, at most 500 iterations were performed by LMNN and 30% of training data were used to do cross-validation. For LMNN-MM, the maximum number of iterations was set to 50 and a step size was fixed to 10^{-7}. For GLML, optimal γ was also achieved via cross-validation. Eventually, the PLML hyper-parameter values were chosen as suggested by [123]. Additionally, α_1 was chosen via cross-validation. For T-R^{2}LML, the value of the regularization parameter λ was cross-validated over $\{10^{-1}, 1, 10^1, ..., 10^6, 10^7\}$. The other parameters values used were set as described in the previous subsection. With respect to E-R^{2}LML, the validation procedure over the set $\{10^{-2}, 10^{-1}, 1, 10^1, 10^2\}$ helped choose the regularization parameter λ. We set the other parameter settings as the same ones used in the previous experiments. Finally, for both methods, K, the number of metrics, was cross-validated over $\{1, 2, ..., 7\}$.

To compare if two methods differ statistically, we employed McNemar's test [76]. Also, note that

[7] `http://www.cse.wustl.edu/~kilian/code/code.html`

[8] `http://www.cs.utexas.edu/~pjain/itml/`

[9] `http://cui.unige.ch/~wangjun/papers/PLML.zip`

Table 3.4: When considering 8 algorithms and 18 datat sets, these are the classification accuracy results. By setting family-wise significance level as 0.05, all the algorithms are ranked from the best to the worst via McNemar's test. If two algorithms do not show statistical difference, they will share the same rank.

	Euclidean	ITML	LMNN	LMNN-MM	GLML	PLML	T-R^{2}LML	E-R^{2}LML
A	65.31^{2nd}	65.86^{2nd}	66.10^{2nd}	66.10^{2nd}	62.28^{3rd}	61.03^{3rd}	58.72^{4th}	$\mathbf{74.16}^{1st}$
B	51.42^{3dr}	$\mathbf{63.92}^{1st}$	$\mathbf{64.73}^{1st}$	$\mathbf{64.73}^{1st}$	57.15^{2nd}	$\mathbf{64.62}^{1st}$	57.19^{2nd}	$\mathbf{66.96}^{1st}$
C	93.15^{2nd}	92.80^{2nd}	93.55^{2nd}	93.70^{2nd}	93.10^{2nd}	$\mathbf{95.55}^{1st}$	93.10^{2nd}	$\mathbf{94.75}^{1st}$
D	87.65^{4th}	91.44^{3rd}	90.13^{3rd}	90.44^{3rd}	91.30^{3rd}	$\mathbf{97.48}^{1st}$	95.03^{2nd}	$\mathbf{96.86}^{1st}$
E	70.02^{3rd}	71.04^{2nd}	70.04^{2nd}	66.80^{2nd}	70.00^{3rd}	$\mathbf{77.44}^{1st}$	$\mathbf{76.89}^{1st}$	$\mathbf{77.61}^{1st}$
F	80.05^{4th}	90.21^{2nd}	90.74^{2nd}	89.42^{2nd}	87.30^{3rd}	90.48^{2nd}	90.16^{2nd}	$\mathbf{92.59}^{1st}$
G	96.51^{2nd}	$\mathbf{96.82}^{1st}$	96.31^{2nd}	96.28^{2nd}	96.49^{2nd}	$\mathbf{97.49}^{1st}$	$\mathbf{97.51}^{1st}$	$\mathbf{97.15}^{1st}$
H	55.95^{5th}	73.72^{3rd}	59.28^{4th}	59.28^{4th}	$\mathbf{97.28}^{1st}$	75.44^{3rd}	80.39^{2nd}	73.51^{3rd}
I	75.57^{3rd}	$\mathbf{86.43}^{1st}$	82.35^{2nd}	82.35^{2nd}	71.95^{3rd}	78.73^{3rd}	$\mathbf{91.86}^{1st}$	$\mathbf{90.50}^{1st}$
J	37.14^{4th}	44.29^{3rd}	$\mathbf{55.71}^{1st}$	47.14^{3rd}	40.00^{4th}	50.00^{3rd}	$\mathbf{54.29}^{1st}$	$\mathbf{58.57}^{1st}$
K	85.94^{4th}	89.70^{2nd}	88.13^{3rd}	89.53^{2nd}	87.34^{3rd}	82.81^{5th}	88.91^{2nd}	$\mathbf{91.56}^{1st}$
L	10.67^{4th}	26.40^{2nd}	15.73^{3rd}	15.73^{3rd}	11.80^{4th}	26.97^{2nd}	$\mathbf{32.58}^{1st}$	$\mathbf{33.34}^{1st}$
M	56.84^{5th}	79.47^{2nd}	77.89^{2nd}	74.21^{3rd}	62.11^{4th}	78.95^{2nd}	$\mathbf{81.05}^{1st}$	$\mathbf{81.05}^{1st}$
N	71.19^{2nd}	$\mathbf{74.10}^{1st}$	$\mathbf{76.08}^{1st}$	$\mathbf{75.78}^{1st}$	70.91^{2nd}	70.25^{2nd}	70.66^{2nd}	72.12^{2nd}
O	89.79^{2nd}	89.33^{2nd}	$\mathbf{93.40}^{1st}$	$\mathbf{93.40}^{1st}$	$89.61^{2}nd$	$88.30^{2}nd$	$\mathbf{91.52}^{1st}$	$\mathbf{92.16}^{1st}$
P	44.53^{4th}	44.53^{4th}	51.36^{2nd}	51.36^{2nd}	39.06^{6th}	42.97^{5th}	$\mathbf{55.47}^{1st}$	48.44^{3rd}
Q	88.09^{3rd}	$\mathbf{90.79}^{1st}$	89.22^{2nd}	89.43^{3rd}	88.45^{3rd}	$\mathbf{90.95}^{1st}$	89.90^{2nd}	$\mathbf{90.79}^{1st}$
R	36.08^{4th}	44.94^{3rd}	39.24^{3rd}	32.91^{4th}	53.17^{2nd}	41.77^{3rd}	$\mathbf{67.72}^{1st}$	55.06^{2nd}

there were 8 algorithms involved in the comparison, we used Holm's step-down procedure. This multiple hypothesis testing method helped control the Family-Wise Error Rate (FWER) [42]. The significance level was set to 0.05. Finally, the experimental results are reported in Table 3.4.

Despite employing a simplistic strategy to infer the weight vector of testing data, E-R^{2}LML achieves the best performance for 14 out of the 18 data sets and outperforms its transductive version, while the other methods outperform E-R^{2}LML on the *Ring Norm*, *Isolet*, *Sonar* and *Wpbc* data sets. GLML's surprisingly good result for the *Ring Norm* data set is probably because GLML assumes the data generates from a Gaussian mixture and the *Ring Norm* is this kind data set. T-R^{2}LML produced best results for 9 out of the 18 data sets. We also notice that T-R^{2}LML achieves almost second best results for the remaining data sets except for *Robot*. For the *Ring Norm*, *Sonar*

and *Wpbc* data sets, T-R^{2}LML even outperforms E-R^{2}LML.

Next, PLML exhibits competitive results, more specifically, best in 6 out of the 18 cases, but performs poorly on some data sets like *COIL20* and *Sonar*, even worse than KNN. For *Glass*, *Heart*, *Isolet* and *Optdigits*, PLML's performance is also quite impressive; it is ranked 2nd among the other methods.

Regarding ITML, although using a global metric, it is ranked first for 5 data sets. Moreover, ITML ranks at least 2nd and performs well especially when dealing with low-dimensional data sets. Finally, GLML performs poorly; Table 3.4 indicates that GLML only achieves 3rd or 4th for 9 out of the 18 data sets.

We can make another observation: metric learning almost always helps improve performance, since Euclidean metric KNN almost always ranked last among all 8 methods considered in Table 3.4. Note that, interestingly, the local metric learning framework, LMNN-MM does not show any significant performance improvement over LMNN which is the global version of the algorithm; As indicated in Table 3.4, LMNN-MM even obtained pooper performance compared to LMNN for some data sets. This might be because that LMNN-MM's performance is degraded by setting the number of classes and the number of local metrics equal. Eventually, according to the obtained results, both T-R^{2}LML and E-R^{2}LML yield much better results for all data sets than LMNN-MM.

Conclusion

In the chapter, in order to improve k-nearest neighbors (KNN) retrieval performance for content-based information retrieval system, a new local metric learning framework is proposed, namely Reduced-Rank Local Metric Learning (R^{2}LML). R^{2}LML learns K Mahalanobis-based local metrics which are combined conically, so that pairs of similar points are measured as being located

close to each other, in contrast to pairs of dissimilar points, for which the opposite is desired. Two variants of the framework were considered: Transductive Reduced-Rank Local Metric Learning (T-R^{2}LML) employs transductive learning to infer the conic combination of metrics to be used for assessing distances between test and training data, while Efficient Reduced-Rank Local Metric Learning (E-R^{2}LML) employs a simpler technique to accelerate the learning process. If T represents the number of iterations, a local analysis of the block-minimization training procedure of both variants has been shown to be convergent at a rate of $\mathcal{O}(1/\sqrt{T})$, which is typical for sub-gradient related algorithms.

In order to show the merits of T-R^{2}LML and E-R^{2}LML, extensive experiments involving 18 benchmark classification problems were performed. First, we studied the effect of regularization in R^{2}LML and showed the importance of the nuclear norm-based regularizer in providing low-rank solutions that avoid over-fitting. Second, the number of local metrics K was varied and its influence on classification accuracy for both T-R^{2}LML and E-R^{2}LML were discussed. We concluded that the number of classes of the data set does not necessarily equal the obtained optimal K. Also, our results indicate that larger data sets do not necessarily require employing a large number of local metrics. Finally, we compared T-R^{2}LML and E-R^{2}LML to several other popular global or local metric learning approaches and demonstrated the superiority of our proposed frameworks.

CHAPTER 4: HASH FUNCTION LEARNING

Introduction

Nowadays, due to the exponentially growth of data, as illustrated in [23], content-based image retrieval (CBIR) has become very popular in the past few years. A CBIR system retrieves similar samples from a large database based on a query sample from a user. By pre-specifying a distance metric, the similarity is evaluated and the nearest neighbors of the query sample are retrieved with respect to this metric. However, in some practical settings, especially when dealing with a large data set, comparing exhaustively with all the samples in the database requires very expensive computations. Moreover, sheer size of each sample may obstruct most CBIR frameworks; for instance, the features from a video or an image may contain more than thousands of dimensions. Moreover, to store all these high-dimensional huge data set also poses a challenge.

If compact binary codes are generated from the original data, the retrieval process can achieve fast similarity search. For example, when binary codes are used, approximate nearest neighbors (ANN) [116] search could achieve sub-liner complexity searching time. Additionally, it is also much more efficient just to store the binary codes. Furthermore, hash functions need to be designed in this criteria: between the data samples in the Hamming space, certain similarity qualities need to be preserved.

Existing popular hashing approaches can be categorized into two families: *data-independent* and *data-dependent*. While the former family designs the hash function based on a non data-driven approach, the latter category, by inferring from data, can be further clustered into supervised, semi-supervised and unsupervised learning tasks.

In this work, we propose a novel hash function learning approach, dubbed *Supervised Hash

Learning (*SHL) (Here, * stands for all three learning scenarios), which exhibits the following advantages: first, in the method, a set of Hamming space codewords is utilized. These learned codewords are used to capture the similarities between the hash codes and group the data. Our framework could also utilize unlabeled data to contribute in the optimization. Additionally, a regularization term is utilized in our framework to move the codewords which represent the same class closer to each other. When some codewords collapse to one single codeword, our framework achieves automatic selection of the codewords. Because of these characteristics, under a single formulation, *SHL is able to engage supervised, unsupervised and semi-supervised hash learning tasks. The latter ability obviously allows the framework to perform transductive hash learning. Note that our framework can be viewed as an Error-Correction Codes (ECOC) method, since both methods consider codewords in their frameworks. Readers can refer to [26] and [79] for more details of ECOC.

In the **Formulation** subsection, we provide *SHL's formulation. In the code space, *SHL attempts to minimize the within-group Hamming distances between data's hash codes and a group's codeword. A kernel-based approach with respect to the hash functions is utilized in *SHL. A new regularization term over codewords is also introduced for *SHL in its formulation. Eventually, the proposed formulation leads to a minimization problem over the Reproducing Kernel Hilbert Space (RKHS) vectors which defines the hash functions and the codewords. Note that, the minimizations over the RKHS vectors derives to several Support Vector Machine (SVM) problems. Thus, each single bit of a sample's hash code is generated by a SVM. Eventually, Multiple Kernel Learning (MKL) method [53] is employed to infer a good kernel.

Next, in **Learning Algorithm**, Majorization Minimization (MM) algorithm is utilized, so that Block Coordinate Descent (BCD) approach can be used to optimize *SHL's framework. To train *SHL, the first block optimization is about training several SVM. The famous sequential solver *LIBSVM* [15] can be utilized here. The MKL parameters are updated in the second block. The

third block involves solving a problem with the non-smooth regularization over codewords, which is optimized by a Proximal Subgradient Descent (PSD). The second and third blocks are computationally fast thanks to closed-form solutions. When confronted with a huge data set, kernel-based methods have a significant computational cost. In this work, a version of our algorithm for big data, which is based on the software *LIBSKYLARK* [106], is also presented.

Finally, in **Experiments**, a series of comparative experiments are conducted. The section emphasizes on supervised CBIR problems. Additionally, we also apply the semi-supervised version of our framework on the foreground/background interactive image segmentation problems. On supervised learning scenarios, when compared with other state-of-art hashing algorithms, the best retrieval accuracy is achieved by *SHL in all the data sets. Theoretical analysis of the method's superior performance are presented in subsection Generalization.

In the following work, $[\cdot]$ denotes the Iverson bracket, *i.e.*, $[\text{predicate}] = 0$, if the predicate is false, and $[\text{predicate}] = 1$, if otherwise. Moveover, matrices and vectors are denoted in boldface. All vectors are column vectors and $\cdot^T$ denotes transposition. Also, we define $\mathbb{N}_K \triangleq \{1, \ldots, K\}$ for any positive integer K.

Related Work

As mentioned in the previous section, there are two families in hashing methods: *data-independent* and *data-dependent*.

Data-independent hashing designs the hashing approaches without the necessity to infer from the data. For instance, in [33], data are projected and thresholded into the Hamming space, by Locality Sensitive Hashing (LSH) randomly to generate compact binary codes. The data is projected onto various hyper-planes, so that similar binary codes may be generated from the closer (in terms of

Euclidean distances) data points. One drawback of LSH is that, normally, it requires a very long bit length to encode enough information for retrieval.

In [55], in order to achieve sublinear time similarity search, the authors combined the metric learning approach with LSH to generate random hash codes. They also formulate a solution for high dimensional data. The final experiments show that their approach performs significantly better than LSH.

In order to address the problem of constructing binary codes for high dimensional data, the authors in [91] proposed a distribution-free encoding approach, which is based on random projections. Data samples are projected, so that the expected Hamming distance between two samples' binary codes is related to a Gaussian Kernel's value. The authors also provide a theoretical analysis of the convergence for the introduced approach. The experiments report their method's superior performance than LSH and some other hashing algorithms.

Data-dependent methods can, in turn, be categorized into supervised, semi-supervised and unsupervised learning paradigms.

The majority of data-dependent hashing approaches has been conducted in the supervised learning scenario. For example, Semantic Hashing [96] is designed to map similar documents to similar binary codes. The framework, which is much faster than LSH, is based on the Restricted Boltzmann Machine (RBM). RBM helps to generate the binary codes. Their experiments of filtering documents show that their approach achieves higher accuracy in retrieval applications than LSH.

Binary Reconstructive Embedding (BRE) was proposed in [56]. In BRE, a cost function, which measures the difference between the distances in the Hamming space and the distances in the original space, is minimized. An efficient coordinate descent algorithm for BRE was also introduced. Unlike the other methods, BRE is easily kernelized and distribution-free. The final experiments

show that BRE outperforms LSH.

In [84], through learning the hash functions from pair-wise side information, Minimal Loss Hashing (MLH) formulates the hashing problems based on a bound from structural Support Vector Machines [136]. They also introduce a type of loss function specifically designed for hashing. Although suffering from a higher computational cost, MLH performs better than BRE and SPH.

In [80], the authors introduced the Label-regularized Max-margin Partition (LAMP) algorithm, which addresses the scenario, where the data set is partially labeled as similar or dissimilar. They formulate the problem as a Constrained Convex-Concave Procedure (CCCP) [138], which is relaxed into a series of convex sub-problems. LAMP can be easily kernelized. Empirical evaluations also validate the superiority of LAMP.

Self-Taught Hashing (STH) [139] first utilizes unsupervised learning to identify binary codes for the documents; Classifiers are trained, in the next step, to generate codes for given queries. In their experiments, Binarized Laplacian Eigenmap (LapEig) and Linear Support Vector Machine (SVM) are utilized. The results show that STH outperforms many other approaches significantly.

LAD, proposed in [110], employs a projection matrix to minimize the in-class covariance and maximize the covariance across classes. Next, the optimal thresholds are computed to transform the projections into binary codes to maximize recognition rates. In essence, their approach performs Linear Discriminant Analysis on the data before binarization. LDAHash achieves exceptional performance compared with the other state-of-art methods.

Boosting-based Hashing were also introduced in [102] and [4]. Parameter-Sensitive Hashing in [102] learns several hash functions, which index examples relating to a particular problem. Their approach actually extends LSH. The experiments on human pose detection show that the introduced method is competitive. Besides, [4] utilizes the weakly labeled positive data to formulate

Adaboost learning framework. Their approach, which was tested on the task of audio retrieval, provides impressive results.

In [60], the authors utilized the triplets of side information to design the hash functions; The relative comparison relationship from the triplets is preserved in their method. The learning algorithm utilizes column generation and, hence, the approach is named CGHash. CGHash generalizes to new data points naturally and provides a convex formulation. The final retrieval performances of CGHash outperform other methods like BRE, STH, MLH, when tested on several popular benchmark data sets.

Kernel Supervised Hashing (KSH) [68] utilizes the equivalence between optimizing the Hamming distances and the binary code inner products. This equivalence helps KSH maximize the Hamming distances on dissimilar pairs and minimize the distances on similar data pairs. Additionally, this equivalence also enables KSH to generate the hash code sequentially and efficiently. The experiments on several popular benchmarks demonstrate KSH's superior performance.

In [62], instead of utilizing kernel functions to achieve non-linearity in hashing, the authors employed boosted decision trees, which are fast to train and evaluate. Their sub-modular formulation is optimized by an efficient GraphCut-based block search. Experiments show that their method outperforms KSH, BRE, STH, both in retrieval accuracy and training time.

The authors of [130] developed a supervised hashing framework for image retrieval, which learns a good image representation. The approach is divided into two stages. In the first stage, in order to optimize over similarity matrix, a coordinate descent method is proposed. In the second stage, the authors employed a deep convolutional network to learn a good feature representation for the images and a set of hash functions. The experiments show that the introduced approach achieves superior performance.

Latent Factor Hashing (LFH) [141] utilizes a latent factor model to learn binary codes which preserve similarity. The authors utilize an algorithm with convergence guarantees to train LFH. Additionally, when dealing with large-scale data set, a stochastic learning framework for the optimization is also proposed.

A framework is presented in [63], which could directly optimize multivariate performance measures such as Area Under Curve (AUC). The authors employed an algorithm which combines cutting plane and column generation algorithms. The framework is so general that it can apply to learning-to-rank and image retrieval. The final results show that it outperforms a few current popular hashing methods such as BRE, STH and CGHash.

There also also many unsupervised hashing approaches proposed: Spectral Hashing (SPH) [129] assumes a uniform data distribution. The hash functions are designed using spectral graph analysis. After relaxing the original problem, the authors formulate a spectral problem which is efficiently solved by graph Laplacian eigenvectors. The authors also show impressive experimental results.

Spectral Embedded Hashing (SEH) in [16] combines SPH with a new regularizer obtained by using a linear regression function. This regularizer is used to control the mismatch between the Hamming codes and the data representation. The authors also proposed Kernel SEH (KSEH) to cope with high-dimensional data. KSEH employs a non-linear regression function to generate the low dimensional data representation. An algorithm to efficiently solve the eigenvalue decomposition problem in both methods is also introduced. Their comprehensive experiments demonstrate the superiority of SEH and KSEH.

Anchor Graph Hashing (AGH) [69] generates compact binary codes by discovering the neighborhood structure in the data. Here, low-rank adjacency matrices are obtained by Anchor Graphs. The authors also describe a hierarchical threshold learning procedure to solve the problem. Finally, the experimental results on two large data sets demonstrate the efficacy of the proposed AGH.

The method in [124] sequentially learns hash functions by correcting the errors from the previous hash function. From the errors in the previous bit, several pseudo-labels are generated by this method. This error is minimized by each new hash function. The superiority of the proposed method is demonstrated by conducting experiments on large data sets.

Iterative Quantization (IQ) was introduced in [37]. By learning an orthogonal matrix, IQ minimizes the quantization error of mapping the data to the binary hypercube's vertices. This method is related to multi-class spectral clustering. The experiments show their approach outperforms LSH and SPH.

The approach in [70] attempts to obtain a subspace shared by the hash functions by dividing the feature space. On one hand, the hash function learning process suppresses the irrelevant information. On the other hand, the final form also takes the complementary subspace into consideration. Finally, the authors designed an objective function combining shared subspace contribution, binary quantization loss and spectral embedding loss. To learn both the hash functions and the shared structure, the authors proposed an efficient alternating optimization method.

The authors of [67] introduced an unsupervised hashing model based on Anchor Graphs [66]. Their method tries to preserve the neighborhood structure in a discrete code space. The authors cast the graph hashing problem into a discrete optimization problem. An alternating maximization algorithm is introduced to cope with the constraints and optimize the framework. The experimental results show that the proposed approach achieves superior results, especially for longer binary codes.

As for semi-supervised hashing, a few approaches were also proposed: Semi-Supervised Hashing (SSH) in [121] and [123] attempts to utilize labeled information to minimize an empirical error. An information theoretic regularizer, which is based on both labeled and unlabeled data, is also considered in SSH to avoid over-fitting.

Another method, semi-supervised tag hashing [125], incorporates tag information into learning hash functions by considering the correlation between hash bits and tags. By ensuring the tag consistency and preserving the image similarities, the hash functions are learned in a unified manner. An iterative coordinate descent algorithm is proposed to optimize the relevant problem. The effectiveness of hashing is also improved by orthogonal transformation by minimizing the quantization error.

[19] proposed Multi-Graph Hashing (MGH), which can effectively combine the multiple modalities with optimized weights in a multi-graph learning scenario. A sequential learning scheme is utilized to learn compact binary codes. The binary codes generated by MGH enables fast similarity search, because the approach enables direct and fast handling for the query examples. Eventually, the experiments show the superiority of the proposed approach.

Finally, note that SVM is also utilized in Self-Taught Hashing (STH) [139] to generate hash codes. However, *SHL and STH are totally different; STH treats the unsupervised and supervised learning tasks as two separate stages. On the other hand, both of the learning paradigms are combined simultaneously in one single cost function of *SHL. Different from STH, *SHL's formulation naturally derives to SVM problems.

Formulation

Hash function learning aims to infer functions generating compact binary codes in a Hamming space from original data samples. Suppose we have the Hamming space $\mathbb{H}^B \triangleq \{-1, 1\}^B$, which means B-bit hash codes. With an arbitrary set $\mathcal{X}$ as sample space, *SHL actually addresses multi-class classification tasks. *SHL learns a hash function $\mathbf{h} : \mathcal{X} \to \mathbb{H}^B$ and a set of $C \times S$ labeled codewords $\boldsymbol{\mu}_{c,s},\ c \in \mathbb{N}_C$ and $s \in \mathbb{N}_S$ (Each class is represented by S codewords). The learning

tries to minimize the distances between the hash code of a labeled sample and the codeword corresponding to the sample's class label. Here distances are measured using the Hamming distance. *SHL also utilizes unlabeled samples to contribute in learning process. Eventually, *SHL classifies a test sample based on the label of the codeword closest to the data sample's hash code.

In our framework, the hash code for a sample $x \in \mathcal{X}$ is finally computed as $\mathbf{h}(x) \triangleq \operatorname{sgn} \mathbf{f}(x) \in \mathbb{H}^B$. Here, the signum function is defined component-wisely. Additionally, $\mathbf{f}(x) \triangleq [f_1(x) \dots f_B(x)]^T$, where $f_b(x) \triangleq \langle w_b, \phi(x) \rangle_{\mathcal{H}_b} + \beta_b$ with $w_b \in \Omega_{w_b} \triangleq \{w_b \in \mathcal{H}_b : \|w_b\|_{\mathcal{H}_b} \leq R_b, R_b > 0\}$ and $\beta_b \in \mathbb{R}$ for all $b \in \mathbb{N}_B$. $\mathcal{H}_b$ is a RKHS with inner product $\langle \cdot, \cdot \rangle_{\mathcal{H}_b}$, induced norm $\|w_b\|_{\mathcal{H}_b} \triangleq \sqrt{\langle w_b, w_b \rangle_{\mathcal{H}_b}}$ for all $w_b \in \mathcal{H}_b$. We also have a associated feature mapping $\phi_b : \mathcal{X} \to \mathcal{H}_b$ and reproducing kernel $k_b : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, such that $k_b(x, x') = \langle \phi_b(x), \phi_b(x') \rangle_{\mathcal{H}_b}$ for all $x, x' \in \mathcal{X}$.

The aforementioned kernel is a symmetric function, $k(\boldsymbol{x}, \boldsymbol{x}') = \phi(\boldsymbol{x})^T \phi(\boldsymbol{x}')$, which is defined based on a non-linear mapping $\phi(\boldsymbol{x})$. By [11], kernel, which is used in large-margin classifiers, was introduced into machine learning. The simplest kernel function can be defined as $\phi(\boldsymbol{x}) = \boldsymbol{x}$, which has $k(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{x}^T \boldsymbol{x}'$. The fact that kernel is formulated as an inner product allows researchers to extend many famous machine learning algorithms using *kernel trick*. The idea is that, if one algorithm is formulated as inner products of input vectors, this product can be replaced with kernels. For example, non-linear PCA [98] can be developed by considering kernel trick in principal component analysis. Other examples include kernelized KNN and Fisher discriminant [94] [78] [5].

Instead of heuristically selecting the kernel functions k_b, MKL, which is introduced in [53], is utilized to infer the feature mapping. Specially, we assume that each RKHS $\mathcal{H}_b$ is formed as the direct sum of M common, pre-specified RKHSs $\mathcal{H}_m$, *i.e.*, $\mathcal{H}_b = \bigoplus_m \sqrt{\theta_{b,m}} \mathcal{H}_m$, where $\boldsymbol{\theta}_b \triangleq [\theta_{b,1} \dots \theta_{b,M}]^T \in \Omega_\theta \triangleq \left\{ \boldsymbol{\theta} \in \mathbb{R}^M : \boldsymbol{\theta} \succeq \mathbf{0}, \|\boldsymbol{\theta}\|_p \leq 1, p \geq 1 \right\}$, $\succeq$ denotes the component-wised relation, $\|\cdot\|_p$ is the usual l_p norm in $\mathbb{R}^M$ and m ranges over $\mathbb{N}_M$. Let us note that, it holds that

$k_b(x, x') = \sum_m \theta_{b,m} k_m(x, x')$ for all $x, x' \in \mathcal{X}$, if each preselected RKHS $\mathcal{H}_m$ has associated kernel function k_m.

Now, suppose we have a training set containing both labeled and unlabeled samples with size N. Respectively, let $\mathcal{N}_L$ and $\mathcal{N}_U$ denote the index sets for these two subsets. Let also l_n for $n \in \mathcal{N}_L$ be the class label of the nth labeled sample. By adjusting the parameters $\boldsymbol{\omega}$, *SHL tries to reduce the following distortion measure

$$E(\boldsymbol{\omega}) \triangleq \sum_{n \in \mathcal{N}_L} \min_s d\left(\mathbf{h}(x_n), \boldsymbol{\mu}_{l_n,s}\right) + \sum_{n \in \mathcal{N}_U} \min_{c,s} d\left(\mathbf{h}(x_n), \boldsymbol{\mu}_{c,s}\right) \tag{4.1}$$

$d(\mathbf{h}, \mathbf{h}') \triangleq \sum_b [h_b \neq h'_b]$, is the Hamming distance. Here, one best codeword of each class will be selected to represent each sample. Note that it is very difficult to directly optimize over the distortion E. Instead, an upper bound $\bar{E}$ of E can be optimized.

Particularly, for a hash code generated by *SHL, we have that $d\left(\mathbf{h}(x), \boldsymbol{\mu}\right) = \sum_b \left[\mu^b f_b(x) < 0\right]$. If one considers $\bar{d}\left(\mathbf{f}, \boldsymbol{\mu}\right) \triangleq \sum_b \left[1 - \mu^b f_b\right]_+$, where $[u]_+ \triangleq \max\{0, u\}$ is the hinge function, we have that $d\left(\operatorname{sgn} \mathbf{f}, \boldsymbol{\mu}\right) \leq \bar{d}\left(\mathbf{f}, \boldsymbol{\mu}\right)$ holds for every $\mathbf{f} \in \mathbb{R}^B$ and any $\boldsymbol{\mu} \in \mathbb{H}^B$. It then holds that

$$E(\boldsymbol{\omega}) \leq \bar{E}(\boldsymbol{\omega}) \triangleq \sum_c \sum_s \sum_n \gamma_{c,n,s} \bar{d}\left(\mathbf{f}(x_n), \boldsymbol{\mu}_{c,s}\right) \tag{4.2}$$

where

$$\gamma_{c,n,s} \triangleq \begin{cases} [c = l_n] \left[s = \arg\min_{s'} \bar{d}\left(\mathbf{f}(x_n), \boldsymbol{\mu}_{l_n,s'}\right)\right] & n \in \mathcal{N}_L \\ \left[(c,s) = \arg\min_{c',s'} \bar{d}\left(\mathbf{f}(x_n), \boldsymbol{\mu}_{c',s'}\right)\right] & n \in \mathcal{N}_U \end{cases} \tag{4.3}$$

Apparently, $\bar{E}$ can be optimized by a three-step Block Coordinate Descent algorithm, which will be described in the next section.

Learning Algorithm

*Algorithm for *SHL*

Via a MM approach [47] and [46], $\bar{E}$ as defined in Eq. (4.2) can be optimized using the next proposition.

Proposition 1. *For any *SHL parameter values $\boldsymbol{\omega}$ and $\boldsymbol{\omega}'$, it holds that*

$$\bar{E}(\boldsymbol{\omega}) \leq \bar{E}(\boldsymbol{\omega}|\boldsymbol{\omega}') \triangleq \sum_c \sum_s \sum_n \gamma'_{c,n,s} \bar{d}\left(\mathbf{f}(x_n), \boldsymbol{\mu}_{c,s}\right) \tag{4.4}$$

where the primed quantities are evaluated on $\boldsymbol{\omega}'$ and

$$\gamma'_{c,n,s} \triangleq \begin{cases} [c = l_n] \left[s = \arg\min_{s'} \bar{d}\left(\mathbf{f}'(x_n), \boldsymbol{\mu}'_{l_n,s'}\right)\right] & n \in \mathcal{N}_L \\ \left[(c,s) = \arg\min_{c',s'} \bar{d}\left(\mathbf{f}'(x_n), \boldsymbol{\mu}'_{c',s'}\right)\right] & n \in \mathcal{N}_U \end{cases} \tag{4.5}$$

Additionally, it holds that $\bar{E}(\boldsymbol{\omega}|\boldsymbol{\omega}) = \bar{E}(\boldsymbol{\omega})$ for any $\boldsymbol{\omega}$. In summa, $\bar{E}(\cdot|\cdot)$ majorizes $\bar{E}(\cdot)$.

Prop. 1 can be proved based on the following fact: for any value of $\gamma'_{c,n,s} \in \{0,1\}$ other than $\gamma_{c,n,s}$ as defined in Eq. (4.3), the value of $\bar{E}(\boldsymbol{\omega}|\boldsymbol{\omega}')$ is always more than $\bar{E}(\boldsymbol{\omega}|\boldsymbol{\omega}) = \bar{E}(\boldsymbol{\omega})$.

With the help of the aforementioned proposition, a MM approach is utilized here. Here, the current estimates of model's parameter are defined as $\boldsymbol{\omega}'$. Note that, by minimizing $\bar{E}(\boldsymbol{\omega}|\boldsymbol{\omega}')$ with respect to $\boldsymbol{\omega}$, improved parameter $\boldsymbol{\omega}^*$ satisfying $\bar{E}(\boldsymbol{\omega}^*) \leq \bar{E}(\boldsymbol{\omega}')$ can be yielded. We could employ BCD to minimize this framework, as will be shown based on the next proposition.

Proposition 2. *Minimizing $\bar{E}(\cdot|\boldsymbol{\omega}')$ with respect to the Hilbert space vectors, the offsets β_p and the MKL weights $\boldsymbol{\theta}_b$, while regarding the codeword parameters as constant, one obtains the following B independent, equivalent problems:*

$$\inf_{\substack{w_{b,m}\in\mathcal{H}_m, m\in\mathbb{N}_M \\ \beta_b\in\mathbb{R}, \boldsymbol{\theta}_b\in\Omega_\theta, \mu^b_{c,s}\in\mathbb{H}}} \lambda_1 \sum_c \sum_s \sum_n \gamma'_{c,n,s} \left[1 - \mu^b_{c,s} f_b(x_n)\right]_+ + \frac{1}{2}\sum_m \frac{\|w_{b,m}\|^2_{\mathcal{H}_m}}{\theta_{b,m}}$$
$$+ \lambda_2 \sum_c \sum_{i,j\in S} \left\|\boldsymbol{\mu}_{c,i} - \boldsymbol{\mu}_{c,j}\right\|_2 \quad b \in \mathbb{N}_B \tag{4.6}$$

where $f_b(x) = \sum_m \langle w_{b,m}, \phi_m(x)\rangle_{\mathcal{H}_m} + \beta_b$ and $\lambda_1 > 0$ is a regularization constant.

By replacing the constraints with regularizations, one can prove this proposition. Eventually, the substitution $w_{b,m} \leftarrow \sqrt{\theta_{b,m}} w_{b,m}$, as illustrated in [53], is also preformed. The third term in Problem (4.6) pushes codewords representing the same class closer to each other. With an appropriate value of λ_2, this regularization helps *SHL automatically select the codewords.

Problem (4.6) is jointly convex with respect to all variables under consideration. Addtionally,

one may recognize it as a binary MKL SVM training problem, which will be showcased in the following proposition.

First block minimization: By considering $w_{b,m}$ and β_b for each b as a single block, one can instead maximize the following problem, instead of optimizing Problem (4.6):

Proposition 3. *The dual form of Problem (4.6) takes the form of*

$$\sup_{\boldsymbol{\alpha}_b \in \Omega_{a_b}} \boldsymbol{\alpha}_b^T \mathbf{1}_{NCS} - \frac{1}{2}\boldsymbol{\alpha}_b^T \mathbf{D}_b[(\mathbf{1}_{CS}\mathbf{1}_{CS}^T) \otimes \mathbf{K}_b]\mathbf{D}_b\boldsymbol{\alpha}_b \quad b \in \mathbb{N}_B \tag{4.7}$$

where $\mathbf{1}_K$ *stands for the all ones vector of* K *elements* ($K \in \mathbb{N}$), $\boldsymbol{\mu}_b \triangleq [\mu_{1,b} \dots \mu_{C,b}]^T$, $\mathbf{D}_b \triangleq \operatorname{diag}(\boldsymbol{\mu}_b \otimes \mathbf{1}_N)$, $\mathbf{K}_b \triangleq \sum_m \theta_{b,m}\mathbf{K}_m$, *where* $\mathbf{K}_m$ *is the data's* m^{th} *kernel matrix,* $\Omega_{a_b} \triangleq \left\{\boldsymbol{\alpha} \in \mathbb{R}^{NC} : \boldsymbol{\alpha}_b^T(\boldsymbol{\mu}_b \otimes \mathbf{1}_N) = 0, \mathbf{0} \preceq \boldsymbol{\alpha}_b \preceq \lambda_1\boldsymbol{\gamma}'\right\}$ *and* $\boldsymbol{\gamma}' \triangleq \left[\gamma'_{1,1,1}, \dots, \gamma'_{1,N,1}, \gamma'_{1,N,2}, \dots, \gamma'_{1,N,S}, \gamma'_{2,N,S}, \dots, \gamma'_{C,N,S}\right]^T$.

The detailed proof is provided in APPENDIX C. Given that $\gamma'_{c,n,s} \in \{0, 1\}$, Problem (4.7) is actually a SVM training problem. Many online software packages, for example, `LIBSVM`, can be used to solve this problem. After SVM training, the quantities $\langle w_{b,m}, \phi_m(x)\rangle_{\mathcal{H}_m}$, β_b and $\|w_{b,m}\|^2_{\mathcal{H}_m}$, which are necessary in the following steps, can be computed.

When dealing with large scale data sets, the sequential solver *LIBSVM* may encounter a memory bottleneck, because of the kernel matrix computations. Therefore a parallel software package, which distributes the kernel matrix computation to various nodes, is necessary for big data problems. *LIBSKYLARK* [106], which utilizes random features [92] to approximate kernel matrix and Alternating Direction Method of Multipliers (ADMM) [88] to parallelize the algorithm, proves to be an efficient solver for large scale SVM problem. *LIBSKYLARK* achieves impressive acceler-

ation, when solving SVM problems compared to *LIBSVM* in [106]. Experiments over large data sets are also conducted.

Second block minimization: We need to optimize MKL parameters $\boldsymbol{\theta}_b$. Fortunately, as illustrated in Prop. 2 of [53] for $p > 1$, the closed-form solution can be utilized here. The closed-form solution is given as

$$\theta_{b,m} = \frac{\|w_{b,m}\|_{\mathcal{H}_m}^{\frac{2}{p+1}}}{\left(\sum_{m'} \|w_{b,m'}\|_{\mathcal{H}_{m'}}^{\frac{2p}{p+1}}\right)^{\frac{1}{p}}}, \quad m \in \mathbb{N}_M, b \in \mathbb{N}_B. \tag{4.8}$$

Third block minimization: we need to optimize this problem due to the new regularization introduced:

$$\inf_{\mu_{c,s}^b \in \mathbb{H}} \sum_n \sum_c \sum_s \gamma'_{c,n,s} \left[1 - \mu_{c,s}^b f_b(x_n)\right]_+ + \lambda_2 \sum_c \sum_{i,j \in S} \left\|\boldsymbol{\mu}_{c,i} - \boldsymbol{\mu}_{c,j}\right\|_2 \tag{4.9}$$

Here, $c \in \mathbb{N}_C, b \in \mathbb{N}_B, s \in \mathbb{N}_S$.

First of all, we relax $\boldsymbol{\mu}$ to continuous values, similar to relaxing the hashcode as continuous when computing the hinge loss. Eq. (4.9) follows the formulation $l(\boldsymbol{x}) + h(\boldsymbol{x})$, which can be solved by proximal methods [89]. Since both the terms (hinge loss and regularization) are convex and non-smooth, we employ PSD method in a similar fashion as in [45], [17] and [93].

The proximal subgradient descent step is

$$\boldsymbol{x}^{k+1} := \mathbf{prox}_{\eta h}(\boldsymbol{x}^k - \eta \partial l(\boldsymbol{x}^k)) \tag{4.10}$$

where η is the step length and ∂l is the subgradient of the function. Here the proximal operator **prox** is defined as:

$$\mathbf{prox}_{\eta h}(\boldsymbol{v}) \triangleq \arg\min_{\boldsymbol{x}} \left(h(\boldsymbol{x}) + \frac{1}{2\eta} \|\boldsymbol{x} - \boldsymbol{v}\|_2^2 \right) \tag{4.11}$$

To obtain the proximal operator, one needs to solve Problem (4.11). In our setting, the regularization term is the sum of many non smooth L_2 norms, whose closed-form proximal operator is not straightforward to derive. Based on the conclusion from [137], the proximal operator of sums of the functions can be approximated by sums of the proximal operator of the individual function, i.e. $\mathbf{prox}_{\sum h} \approx \sum \mathbf{prox}_h$. Thus, all we need is the closed-form proximal operator for one individual norm in Eq. (4.9), *i.e.* $\sum_{i,j \in S} \left\| \boldsymbol{\mu}_i - \boldsymbol{\mu}_j \right\|_2$. Let us concatenate all codewords as $\boldsymbol{\mu} = \left[\boldsymbol{\mu}_1^T, \cdots, \boldsymbol{\mu}_S^T \right]^T \in \mathbb{R}^{BS}$. Moreover, let us define a vector: $\boldsymbol{o} \triangleq [0, \cdots, 1, \cdots, -1, \cdots, 0] \in \mathbb{R}^S$, where the value for index i is set to 1 and -1 for index j. With the definition of a matrix $\boldsymbol{U} \triangleq \boldsymbol{o} \otimes \boldsymbol{I}_B \in \mathbb{R}^{B \times BS}$, the regularization can be reformulated as $h(\boldsymbol{\mu}) = \|\boldsymbol{U}\boldsymbol{\mu}\|_2$, whose proximal operator will be given in the following proposition:

Proposition 4. *Given the norm as:* $h(\boldsymbol{\mu}) = \|\boldsymbol{U}\boldsymbol{\mu}\|_2$. *Following the definition in Eq. (4.11), the proximal operator of this norm:*

Algorithm 2 Algorithm for Problem (4.6)

Input: Training Samples X, Bit Length B.
Output: $\boldsymbol{\omega}$.
1. Initialize $\boldsymbol{\omega}$.
2. **While Not Converged**
3. **For each bit**
4. $\gamma'_{g,n,s} \leftarrow Eq.\ (4.5)$.
5. Step 1: Update $w_{b,m}$ and β_b.
6. Step 2: Compute $\|w_{b,m}\|^2_{\mathcal{H}_m}$.
7. Update $\theta_{b,m}$.
8. Step 3: **For** $k = 1, 2, \cdots$ **do**
9. $\boldsymbol{z}^k = \boldsymbol{\mu}^k - \eta \partial l(\boldsymbol{\mu}^k)$.
10. $\boldsymbol{y}^k = \sum \mathbf{prox}_{\eta h}(\boldsymbol{z}_k)$.
11. $\boldsymbol{\mu}^{k+1} = \boldsymbol{y}^k + \frac{k-1}{k+2}(\boldsymbol{y}^k - \boldsymbol{y}^{k-1})$.
12. **End For**
13. **End For**
14. **End While**
15. Output $\boldsymbol{\omega}$.

$$\boldsymbol{prox}_{\eta h}(\boldsymbol{v}) = \begin{cases} \boldsymbol{\mu}_1 = \boldsymbol{v}_1 \\ \quad \vdots \\ \boldsymbol{\mu}_i = \alpha_1 \boldsymbol{v}_i + \alpha_2 \boldsymbol{v}_j \\ \quad \vdots \\ \boldsymbol{\mu}_j = \alpha_2 \boldsymbol{v}_i + \alpha_1 \boldsymbol{v}_j \\ \quad \vdots \\ \boldsymbol{\mu}_S = \boldsymbol{v}_S \end{cases} \tag{4.12}$$

where $\alpha_1 \triangleq 1 - \alpha_2$ *and* $\alpha_2 \triangleq \min\{\frac{\eta}{\|\boldsymbol{v}_i - \boldsymbol{v}_j\|_2}, \frac{1}{2}\}$, $\boldsymbol{v} \triangleq \left[\boldsymbol{v}_1^T, \cdots, \boldsymbol{v}_S^T\right]^T$, *which is the input vector for proximal operators in Eq. (4.11).*

The detailed proof of Prop. 4 is showcased in APPENDIX D.

Note that, if we consider only one codeword for each class, Problem (4.9) can be simplified without the regularization:

$$\inf_{\mu_c^b \in \mathbb{H}} \sum_n \sum_c \gamma'_{c,n} \left[1 - \mu_c^b f_b(x_n)\right]_+ \tag{4.13}$$

Problem (4.13) can be optimized by mere substitution like in [44].

In conclusion, as depicted in Algorithm 2, *SHL's algorithm contains one MKL update and one SVM optimization for each bit. For the third step, we evaluate the proximal operator for each regularization and compute the summation. After this, the algorithm optimizes codewords using PSD. $\gamma'_{c,n,s}$ is then updated according to the current estimate of the parameters. This algorithm, as shown in Algorithm 2, continues running until reaching the convergence. Per iteration, our algorithm achieves a complexity of $\mathcal{O}(BN^3)$, since `LIBSVM` features a $\mathcal{O}(N^3)$ complexity [64]. Note that N is the number of instances and B is the code length.

Insights to Generalization Performance

In the next section, the comparison experiments with other state-of-the-art hash function learning approaches demonstrate the superiority of *SHL. By noticing that *SHL tries to minimize the Hamming distance between the correct codeword and a labeled sample, this impressive performance can be explained to some extent. For reasons of convenience, we only consider the training set with only labeled samples (*i.e.*, $N = \mathcal{N}_L, \mathcal{N}_U = 0$). Additionally, we only consider one single codeword *SHL. The definitions of the MKL hypothesis space for *SHL can be found in **Formu-**

lation subsection. Before we provide the generalization results, the following two definitions are necessary.

Definition 2. *A Random Variable (RV) σ that is Bernoulli distributed such that $Pr\{\sigma = \pm 1\} = \frac{1}{2}$ will be called a Rademacher RV.*

Definition 3. *Let $\mathcal{G} \triangleq \{g : \mathcal{Z} \mapsto \mathbb{R}\}$ be a set of functions of an arbitrary domain $\mathcal{Z}$ and $Q = \{z_n\}_{n=1}^N$ a set of iid samples from $\mathcal{Z}$ according to a distribution D. Then, the Empirical Rademacher Complexity (ERG) of $\mathcal{G}$ w.r.t Q is defined as:*

$$\widehat{\Re}_Q(\mathcal{G}) \triangleq \mathbb{E}_{\boldsymbol{\sigma}} \left\{ \sup_{g \in \mathcal{G}} \frac{1}{N} \sum_{n=1}^{N} \sigma_n o(z_n) \middle| Q \right\} \tag{4.14}$$

where $\mathbb{E}_{\boldsymbol{\sigma}} \{\cdot\} \triangleq \mathbb{E}_{\sigma_1} \{\mathbb{E}_{\sigma_2} \{\cdots \mathbb{E}_{\sigma_N} \{\cdot\} \cdots\}\}$ and $\{\sigma_n\}_{n=1}^N$ are iid Rademacher RV. In the rest of the section, the condition on Q will be omitted for simplicity. Additionally, the Rademacher Complexity (RC) of $\mathcal{G}$ for a sample size N is defined as:

$$\Re_N(\mathcal{G}) \triangleq \mathbb{E}_{Q \sim D^N} \left\{ \widehat{\Re}_Q(\mathcal{G}) \right\} \tag{4.15}$$

We also need the following two lemmas before showing our final concentration results.

Lemma 1. *Let $\mathcal{Z}$ be an arbitrary set, $\widetilde{\mathcal{F}} \triangleq \{\mathbf{f} : z \mapsto \mathbf{f}(z) \in \mathbb{R}^B,\ z \in \mathcal{Z}\}$, $\Psi : \mathbb{R}^B \to \mathbb{R}$ be L-Lipschitz continuous w.r.t $\|\cdot\|_1$. Also, define $\Psi \circ \widetilde{\mathcal{F}} \triangleq \{g : z \mapsto \Psi(\boldsymbol{f}(z))\}$ and $\left\|\widetilde{\mathcal{F}}\right\|_1 \triangleq \{h : z \mapsto \|\boldsymbol{f}(z)\|_1,\ \boldsymbol{f} \in \widetilde{\mathcal{F}}\}$ then*

$$\widehat{\Re}_Q\left(\Psi \circ \widetilde{\mathcal{F}}\right) \leq L\widehat{\Re}_Q\left(\left\|\widetilde{\mathcal{F}}\right\|_1\right) \tag{4.16}$$

where Q is a set of N samples drawn from $\mathcal{Z}$.

Lemma 2. *Let $\mathcal{Z}$ be an arbitrary set. Define: $\widetilde{\mathcal{F}} \triangleq \{\boldsymbol{f} : z \mapsto \boldsymbol{f}(z) \in \mathbb{R}^B,\ z \in \mathcal{Z}\}$, $\left\|\widetilde{\mathcal{F}}\right\|_1 \triangleq \{h : z \mapsto \|\boldsymbol{f}(z)\|_1,\ \boldsymbol{f} \in \widetilde{\mathcal{F}}\}$ and $\mathbf{1}^T\widetilde{\mathcal{F}} \triangleq \{g : z \mapsto \mathbf{1}^T\boldsymbol{f}(z),\ \boldsymbol{f} \in \widetilde{\mathcal{F}}\}$. Let's further assume that if $\boldsymbol{f}(z) \triangleq [f_1(z), ..., f_B(z)]^T \in \widetilde{\mathcal{F}}$ for $z \in \mathcal{Z}$, then also $[\pm f_1(z), ..., \pm f_B(z)] \in \widetilde{\mathcal{F}}$ for any combination of signs. Then:*

$$\widehat{\Re}_Q\left(\left\|\widetilde{\mathcal{F}}\right\|_1\right) \leq \widehat{\Re}_Q\left(\mathbf{1}^T\widetilde{\mathcal{F}}\right) \tag{4.17}$$

where Q is a set of N samples drawn from $\mathcal{Z}$.

The detailed proof is provided in APPENDIX E for Lemma 1 and in APPENDIX F for Lemma 2.

The following sets of functions are necessary to show the main theoretical analysis

$$\bar{\mathcal{F}} \triangleq \{\mathbf{f} : x \mapsto [f_1(x), ..., f_B(x)]^T,\ f_b \in \mathcal{F},\ b \in \mathbb{N}_B\} \tag{4.18}$$

$$\mathcal{F} \triangleq \{f : x \mapsto \langle w,\ \phi_{\boldsymbol{\theta}}(x)\rangle_{\mathcal{H}_{\boldsymbol{\theta}}} + \beta,\ \ \beta \in \mathbb{R},\ \boldsymbol{w} \in \Omega_w(\boldsymbol{\theta}),\ \boldsymbol{\theta} \in \Omega_{\boldsymbol{\theta}}\} \tag{4.19}$$

where $\Omega_w(\boldsymbol{\theta}) \triangleq \{w \in \mathcal{H}_{\boldsymbol{\theta}} :\ \|w\|_{\mathcal{H}_{\boldsymbol{\theta}}} \leq R\}$ for some $R \geq 0$ and $\Omega_{\boldsymbol{\theta}} \triangleq \{\boldsymbol{\theta} \in \mathbb{R}^M :\ \boldsymbol{\theta} \succeq \mathbf{0},\ \|\boldsymbol{\theta}\|_p \leq 1\}$ for some $p \geq 1$.

Theorem 4. *Assume reproducing kernels of* $\{\mathcal{H}_m\}_{m=1}^M$ *s.t.* $k_m(x,x') \leq r^2,\ \forall x,x' \in \mathcal{X}$ *and a set* Q *of iid samples* $Q = \{(x_1,l_1),...,(x_N,l_N)\}$*. Then for* $\rho > 0$ *independent of* Q*, for any* $\mathbf{f} \in \bar{\mathcal{F}}$*, any* $\boldsymbol{\mu} \in \mathcal{M}$*, where* $\mathcal{M} \triangleq \{\boldsymbol{\mu} :\ \mathbb{N}_C \to \mathbb{H}^B\}$ *and any* $0 < \delta < 1$*, with probability* $1-\delta$*, it holds that:*

$$er\left(\mathbf{f},\boldsymbol{\mu}\right) \leq \widehat{er}\left(\mathbf{f},\boldsymbol{\mu}\right) + \frac{2R}{\rho}\sqrt{rM^{\frac{1}{p'}}\sqrt{\frac{p'}{N^3}}} + \sqrt{\frac{\log\left(\frac{1}{\delta}\right)}{2N}} \tag{4.20}$$

where $er\left(\mathbf{f},\boldsymbol{\mu}\right) \triangleq \frac{1}{B}\mathbb{E}\{d\left(\operatorname{sgn}\mathbf{f}(x),\boldsymbol{\mu}(l)\right)\}$*,* $l \in \mathbb{N}_C$ *is the true label of* $x \in \mathcal{X}$*,* $\widehat{er}\left(\mathbf{f},\boldsymbol{\mu}_l\right) \triangleq \frac{1}{NB}\sum_{n,b}\Phi_\rho\left(f_b(x_n)\mu_{l_n,b}\right)$*, where* $\Phi_\rho(u) \triangleq \min\left\{1,\max\left\{0,1-\frac{u}{\rho}\right\}\right\}$ *and* $p' \triangleq \frac{p}{p-1}$*.*

The detailed proof for Theorem 4 can be found in APPENDIX G.

Experiments

Results of Supervised Hash Learning

In this section, *SHL is compared with other state-of-the-art hashing algorithms: namely Spectral Hashing (SPH)[1] [129], Locality-Sensitive Hashing (LSH) [33], single-layer Anchor Graph Hashing (1-AGH) (2-AGH)[2], Binary Reconstructive Embedding (BRE)[3] [56] and [69]Kernel Supervised Learning (KSH)[4] [68]. These methods are widely used as baseline algorithms in the other

[1] `http://www.cs.huji.ac.il/~yweiss/SpectralHashing/sh.zip`

[2] `http://www.ee.columbia.edu/ln/dvmm/downloads/WeiGraphConstructCode2011/dlform.htm`

[3] `http://web.cse.ohio-state.edu/~kulis/bre/bre.tar.gz`

[4] `http://www.ee.columbia.edu/ln/dvmm/downloads/WeiKSHCode/dlform.htm`

hash function learning papers.

Five data sets, which are widely utilized in other hashing papers as benchmarks, were considered:

- *Pendigits*: a digit data set (10 classes, 256 features and $10,992$ samples) of 44 writers from the *UCI Repository*[5]. In our experiment, $3,000$ for training are randomly chosen and the rest for testing.
- *USPS*: a digit data set also from the *UCI Repository*, is numeric data from the scanning handwritten digits on the envelops provided by the United States Postal Service. Among the data set (10 classes, 256 features and $9,298$ samples), again, we chose 3000 for training and others for testing.
- *Mnist*[6]: a hand written digit data set which contains $70,000$ samples, 784 features and 10 classes. The digits have been centered and size-normalized. In our experiment, we employ $6,000$ samples for training.
- *CIFAR-10*[7]: a labeled image data set collected from *80 million tiny images* [8]. The data set consists of $60,000$ samples, 10 classes, $1,024$ features. Again, $6,000$ samples are used for training and the rest for testing.
- *PASCAL07*[9] [29]: a data set contains annotated consumer photographs collected from the flickr photo sharing website. The data set has 10 classes and 6878 samples. Each sample contains $1,024$ features. Here, we utilize $3,000$ samples for training.

[5] `http://archive.ics.uci.edu/ml/`

[6] `http://yann.lecun.com/exdb/mnist/`

[7] `http://www.cs.toronto.edu/~kriz/cifar.html`

[8] `http://groups.csail.mit.edu/vision/TinyImages/`

[9] `http://pascallin.ecs.soton.ac.uk/challenges/VOC/`

We run all the algorithms for 5 times and report the average performances. Two measurements are considered: (i) k-closest retrieval precisions; we used $k = \{10, 15, \ldots, 50\}$. (ii) Precision-Recall (PR) curves. Within a Hamming radius of $r \in \mathbb{N}_B$, both retrieval recall and precision are computed for hash codes.

For *SHL, parameters are set as follows: the regularization parameter for SVM, λ_1 was set to 1000 since this gives best results when considering these data sets; For MKL, we utilized 11 kernels: 1 polynomial kernel, 1 linear kernel and 9 Gaussian kernels. Both the polynomial kernel and linear kernel were normalized. We also set the bias to 1.0 for the polynomial kernel and its degree was fixed to 2. For the 9 Gaussian kernels, we used the following bandwidth σ values: $[2^{-7}, 2^{-5}, 2^{-3}, 2^{-1}, 1, 2^1, 2^3, 2^5, 2^7]$. All these kernels' parameters are set following the other MKL papers. $p = 2$ for the MKL constraint set. λ_2 was set to 2000 for *Pendigits*, *USPS* and *PASCAL07* and $\lambda_2 = 6000$ for the rest of the data sets.

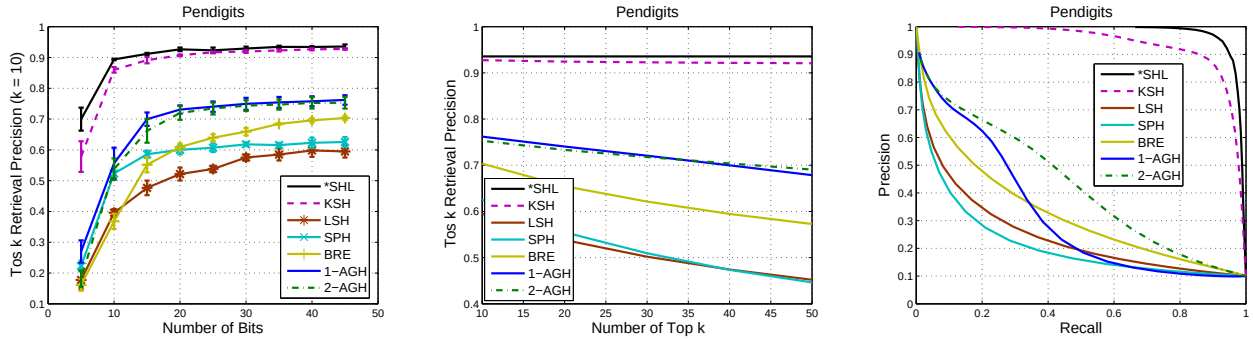


Figure 4.1: For *Pendigits*, the top k retrieval results and Precision-Recall curve for all the methods considered.

For the other state-of-art hashing algorithms, namely SPH, BRE, AGH and KSH, parameters were used as suggested by the authors. We report all the results in Figure 4.1, Figure 4.2, Figure 4.3, Figure 4.4 and Figure 4.5.

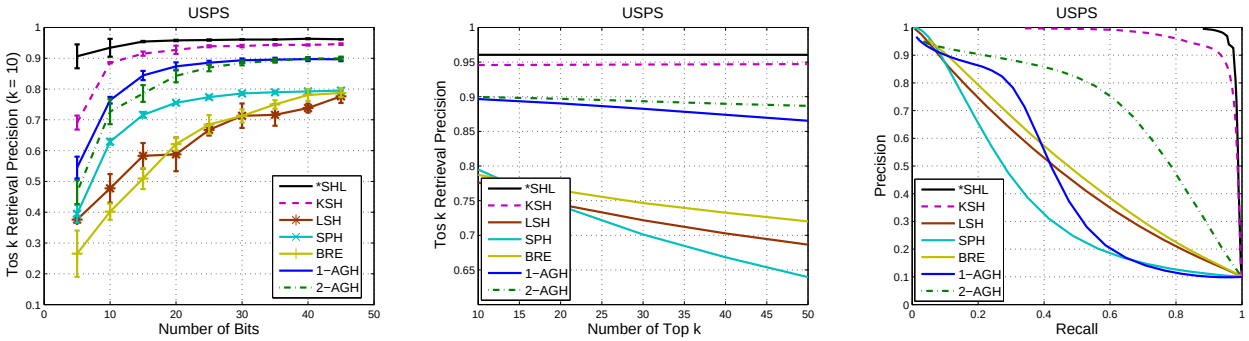


Figure 4.2: For *USPS*, the top k retrieval results and Precision-Recall curve for all the methods considered.

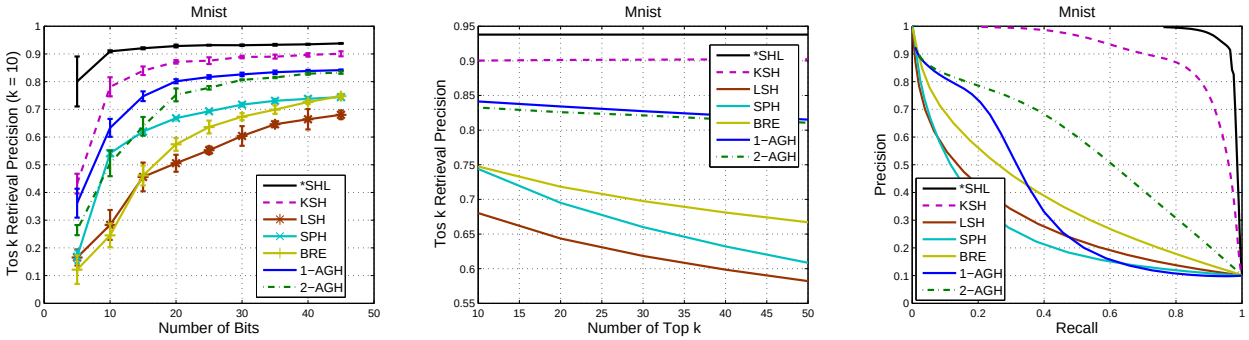


Figure 4.3: For *Mnist*, the top k retrieval results and Precision-Recall curve for all the methods considered.

Apparently, our proposed framework, *SHL, achieves the best performance among the algorithms KSH, LSH, SPH, BRE and AGH. For all the data sets, *SHL offers the best top-10 retrieval precision results. *SHL achieves significant results especially for non-digits data sets (*CIFAR-10*, *PASCAL07*). When utilizing PR-curve as performance metric, the largest areas under the curve are also achieved by *SHL. Although, as reported in [68], KSH achieves impressive results, in our ex-

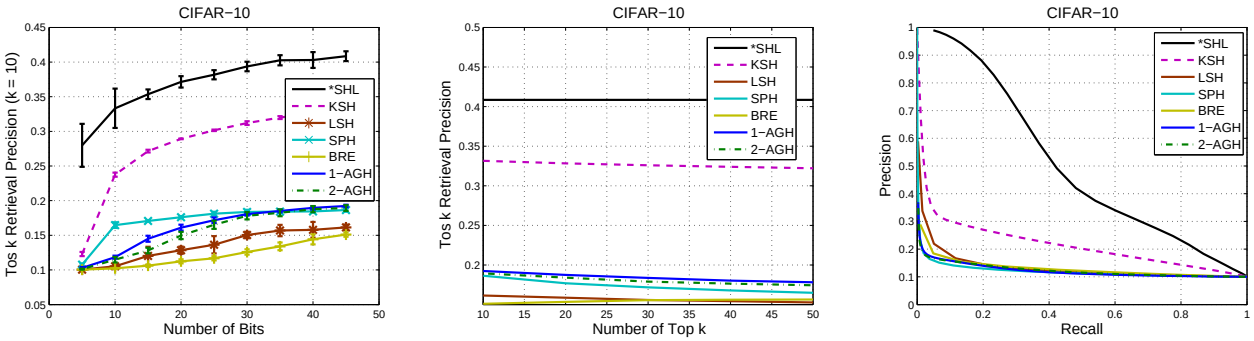


Figure 4.4: For *CIFAR-10*, the top k retrieval results and Precision-Recall curve for all the methods considered.

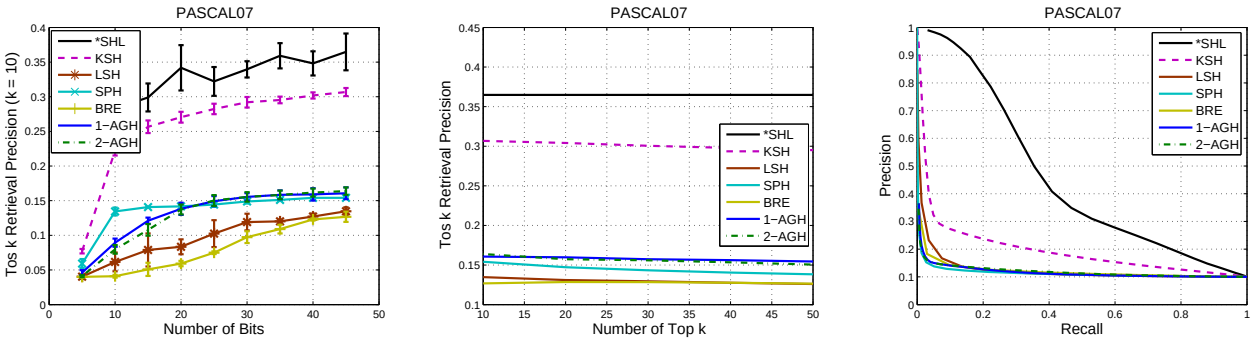


Figure 4.5: For *PASCAL07*, the top k retrieval results and Precision-Recall curve for all the methods considered.

periments, *SHL outperforms KSH across all data sets. Additionally, except BRE, we notice that supervised hashing outperforms unsupervised hashing frameworks. When considering a longer bit length, BRE may achieve better performance like in Figure 4.1 and Figure 4.3. Moreover, it is worth mentioning that *SHL performs impressively with short bit length, since we can observe an obvious gap between the curves. When considering longer bits, *SHL's superiority fades away.

Compared with other unsupervised hashing algorithms, AGH also yields good results, because it utilizes anchor points as side information to produce hash codes. When dealing with the non-digits data sets we considered (*CIFAR-10* and *PSACAL07*), except KSH adn *SHL, the other algorithms perform poorly.

In terms of top-k retrieval precision, the other methods' performance degraded when the top-k number is varied from 10 to 50, except *SHL and KSH. For the other approaches, when retrieving more items, the method may generate poorer performance. For KSH and *SHL, the kernel methods help better capture the similarities between data samples, which gives robust retrieval performance. We observe that KSH performs slightly worse for *CIFAR-10* and *PSACAL07*, when k increases, while the performance of *SHL still remains robust. Note that, in image of digits data sets, the robustness of the two-layer AGH is better than its single-layer version. Eventually, in Figure 4.7 and Figure 4.6, the qualitative results for the *Mnist* and *CIFAR-10* data sets are showcased. In conclusion, for every code length we considered, it seems that *SHL's performances are superior.

Transductive Hash Learning Results

This subsection compares *SHL's inductive version's performance with its transductive [120] learning version. To the best of our knowledge, *SHL is the only hash learning approach to date that can incorporate transductive learning. In this experiment, the *Letter* and *Vowel* from *UCI Machine Learning Repository* are used. For the *Vowel*, 330 samples were randomly chosen for training and 220 for testing. As for the *Letter* data set, 300 training samples are randomly chosen. We run 20 times for each learning scenario and we vary the code length (B) from 4 to 15 bits. Figure 4.8 reports the results, revealing the transductive *SHL learning's potential advantages.

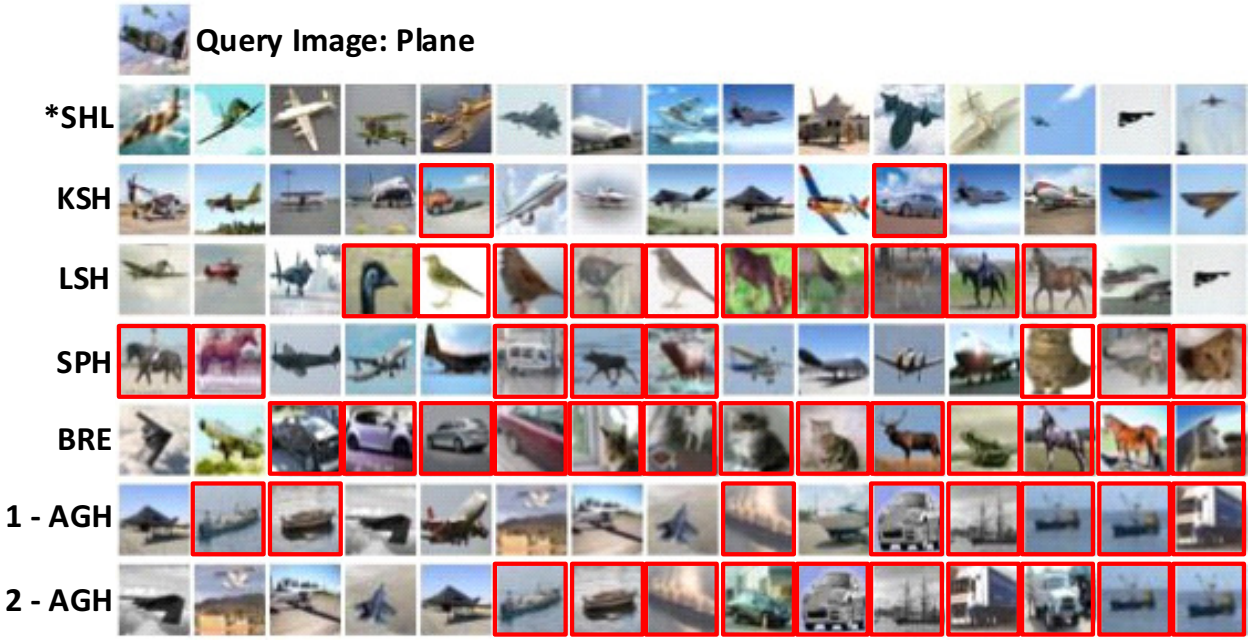


Figure 4.6: For *CIFAR-10*, retrieval results on ”Plane”. For each algorithm, we use 45-bit binary codes to retrieve 15 images. Red box indicates wrong retrieval results.

Image Segmentation

Besides content-based image retrieval, the proposed *SHL can also be utilized in other scenarios, for example, the foreground/background interactive image segmentation [10], where the images are partially labeled as foreground and background by users. In *SHL, while foreground and background are represented by two codewords, the rest of the pixels can be labeled in semi-supervised learning scenario. This subsection shows the interactive image segmentation results using *SHL on the data set introduced in [38]. The hash code length is 5 and the rest of the parameters settings follow the previous section. For each pixel, the RGB values are used as features. The results are shown in Figure 4.9. We notice that, provided with partially labeled information, *SHL successfully segments the foreground object from the background. Especially in (e), although all the

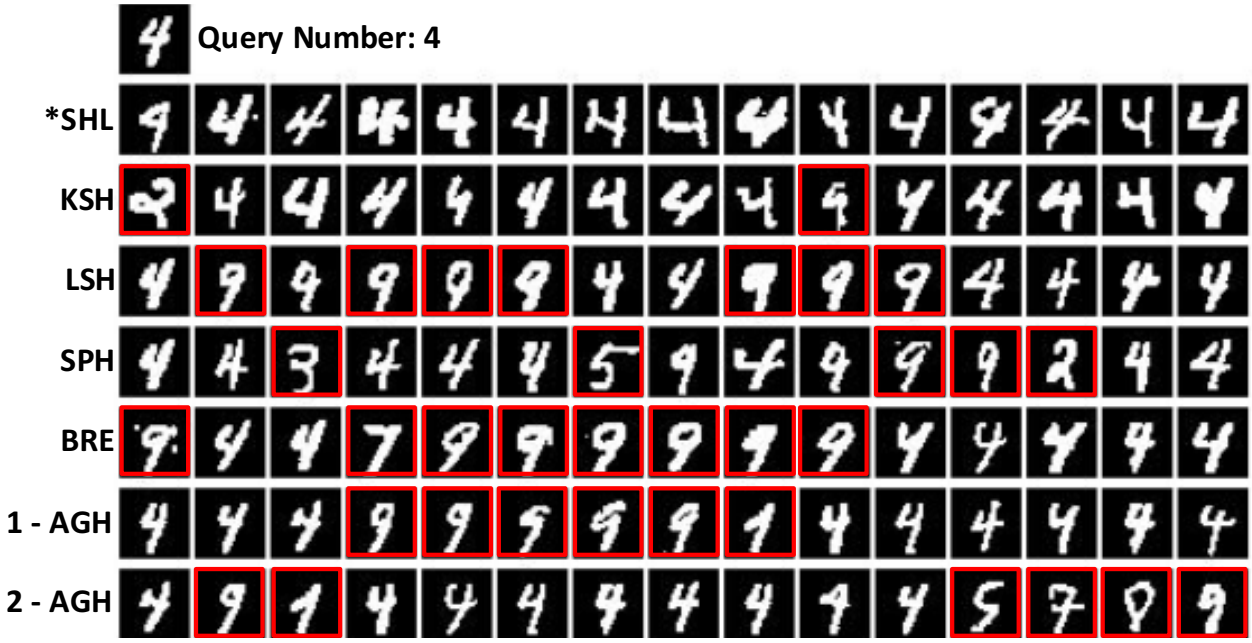


Figure 4.7: For *Mnist*, retrieval results on ”4”. For each algorithm, we use 45-bit binary codes to retrieve 15 images. Red box indicates wrong retrieval results.

flower pots share the same color, *SHL only highlight the labeled one and its plant. Additionally, for some images, like (c) and (f), shaded areas fail to be segmented. In these cases, *SHL may need more image features to generate better segmentation results.

**SHL for Large Data Set*

Suppose *SHL is used to cope with a huge date set, whose kernel matrix might be so large that can not be fit into the memory of one single machine. Therefore, instead of using the sequential solver *LIBSVM*, we need the parallel solver *LIBSKYLARK* [10] for *SHL. *LIBSKYLARK* decomposes the huge data set into smaller data chunks, which will be processed in each node of a large cluster.

[10] `http://xdata-skylark.github.io/libskylark/`

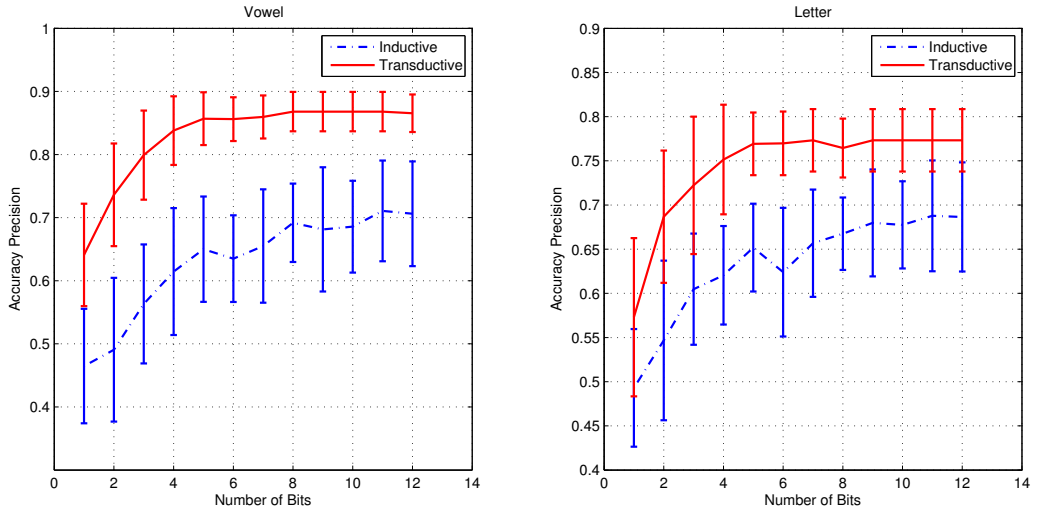


Figure 4.8: Comparison results between Inductive learning and Transductive Learning.

In order to show *SHL's capability of solving big data set, we created clusters in *Amazon Web Service*[11]. Two clusters were created with one containing 2 nodes while the other includes 10 nodes. Each node has Xeon E5-2666 CPU and 3.75 GB memory.

Three data sets are considered here:

- *USPS*: with 256 features, 7291 samples are used for training, while 2007 samples are for testing.
- *Mnist*: 60K are for training and 10K are for testing. This data set contains 784 features.
- *Mnist1M*: 784 features. Here, we train the model using 1 million samples. For testing, 100K samples are used.

[11] `https://aws.amazon.com/`

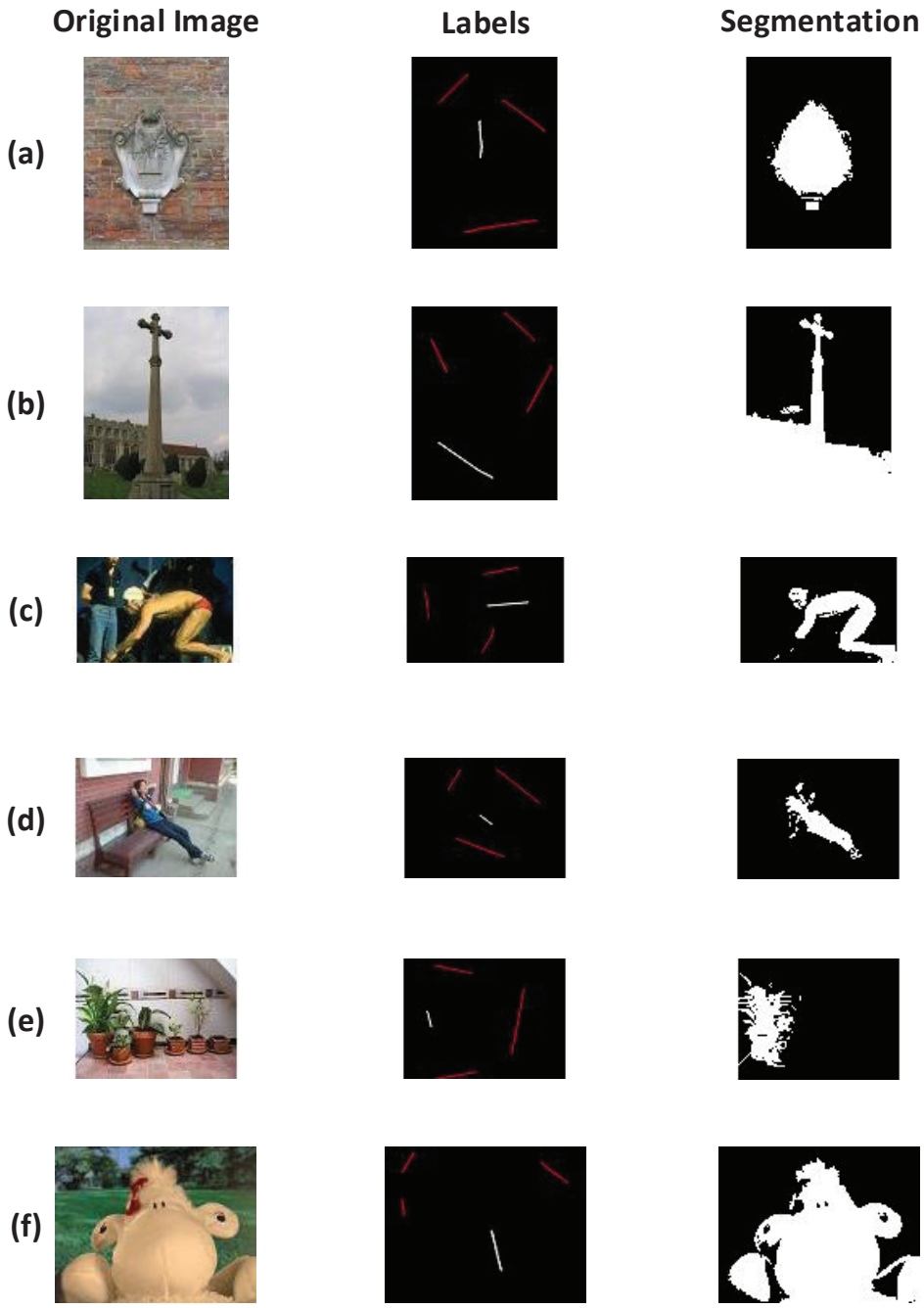


Figure 4.9: Foreground/Background interactive image segmentation. The left column contains the original images. The middle column includes labeled pixels. The right column shows the results of the segmentati on.

We run the experiments using 5 bits, 25 bits and 45 bits of the codeword for the three data sets. *SHL's parameters are set as the previous sections. The results of running time and top-10 retrieval accuracies are reported in Table 4.1:

Several observations can be made here: firstly, *LIBSKYLARK* provides *SHL a faster training process. As reported in Table 4.1, *SHL only needs about 7 minutes to train *USPS* when considering 45-bits binary codes. Moreover, although *LIBSKYLARK* utilizes random features to approximate kernel matrix, the retrieval accuracy is still competitive. Secondly, larger cluster is obviously more powerful. Here, *Mnist* run on the 10-node cluster needs almost 25% running time compared with the same data set on a 2-node cluster. Both clusters provide similar and competitive retrieval accuracies. Finally, when given a large data set like *Mnist1M*, *SHL with *LIBSKYLARK* could achieve reasonable results using about 14 hours training time. Therefore, *SHL is able to solve big data problems utilizing *LIBSKYLARK*.

Conclusions

In this part, a novel hash learning framework was proposed, namely *SHL. The method has the following main advantages: first, the training algorithm is simple and efficient to implement. Secondly, the framework can address supervised, semi-supervised and unsupervised learning scenarios. Additionally, after introducing a regularization over the multiple codewords, we also provide the PSD method to solve this regularization.

Eventually, experiments on 5 benchmark data sets were conducted. A comparison between our methods with the other 6 popular hashing approaches shows *SHL to be very impressive and highly competitive. Moreover, we also give results on transductive learning scenario. Additionally, another application based on our framework, interactive image segmentation, is also showcased in

Table 4.1: Top-10 retrieval results and running time for *SHL for various data sets. Here, running time secs means seconds, mins means minutes and hours means hours.

DATA	NODES	TRAINING	TESTING	BITS	ACCURACY	TIME
				5	0.916	45.82 $secs$
USPS	2	7291	2007	25	0.936	227.17 $secs$ / 3.78 $mins$
				45	0.941	408.66 $secs$ / 6.90 $mins$
				5	0.826	1690.46 $secs$ / 28.17 $mins$
Mnist	2	$60K$	$10K$	25	0.960	8406.04 $secs$ / 140.08 $mins$
				45	0.969	4253.28 $secs$ / 251.85 $mins$
				5	0.839	487.88 $secs$ / 8.13 $mins$
Mnist	10	$60K$	$10K$	25	0.962	2361.07 $secs$ / 39.35 $mins$
				45	0.964	4253.28 $secs$ / 70.89 $mins$
				5	0.824	7918.38 $secs$ / 131.97 $mins$
Mnist1M	10	$1M$	$100K$	25	0.927	28066.37 $secs$ / 467.77 $mins$
				45	0.935	48574.96 $secs$ / 809.58 $mins$

the experimental section. Finally, we show that *SHL can also cope with huge data sets.

CHAPTER 5: CONCLUSION AND FUTURE WORKS

This dissertation focuses on information retrieval system. Content-based information retrieval, which retrieves the similar items based on the features extracted from the information, has become our main research focus. The most popular and important algorithm utilized in information retrieval system is k-nearest neighbors (KNN) and our research aims to improve KNN's retrieval accuracy and its retrieval speed.

Firstly, Distance Metric Learning (DML) has been studied to improve KNN search accuracy, which is based on the Euclidean distance. After many researchers proposed various DML approaches, we contributed to this research field a new local metric learning framework, dubbed Reduced-Rank Local Metric Learning (R^2LML). R^2LML considers a conic combination of several local metrics. A regularization term is also incorporated to suppress the noise and avoid over-fitting. Two formulations of DML are introduced in the dissertation. Transductive Reduced-Rank Local Metric Learning (T-R^2LML) employs transductive learning and has heavy computation burden. Efficient Reduced-Rank Local Metric Learning (E-R^2LML), on the other hand, has a faster training procedure. A theoretical convergence analysis of the proposed block coordinate algorithms for both the frameworks is also provided. Finally, we conducted extensive experiments. The first experiment validates the effectiveness and usefulness of the regularization, which is used to avoid over-fitting and achieve sparse solutions. The second experiment shows that there is no obvious relationship between R^2LML's performance and the number of local metrics. Eventually, the comparison between both our approaches and the other popular metric learning algorithms shows the superiority of T-R^2LML and E-R^2LML.

On the other hand, in order to accelerate the retrieval process via KNN in content-based information retrieval system, Hash Function Learning (HFL) was also studied. The model attempts to

learn several hash functions, which generate compact binary codes from the original data samples. If, in the original space, two points are similar, the binary code of them should also lie closer to each other in the Hamming space. We proposed a new HFL framework, *Supervised Hash Learning (*SHL), which could successfully incorporate supervised, semi-supervised and unsupervised learning scenarios. A block coordinate descent algorithm based on Majorization Minimization (MM) technique is also introduced to optimize the framework. We also provide a theoretical concentration bound on our proposed algorithm. Finally, in the experimental comparisons, *SHL outperforms other popular hash function learning algorithms. Additionally, in order to tackle large data sets, *SHL was also shown to solve big data problems utilizing parallel computing technique. In the experiments, we also showed that our framework is capable of dealing with large data sets with the help of an on-line software, *LIBSKYLARK*.

The idea of codewords in HFL can be further extended. For instance, codewords can be employed in multi-label learning [25], in which class labels are vectors. Each bit of class label represents one object in data sets. For example, in image annotation, each bit of class label represents one object in the image. One image can be labeled containing sky, ocean and cloud at the same time. Suppose we define a pair of codewords $\boldsymbol{\mu}^+$ and $\boldsymbol{\mu}^-$ for one bit. The previous codeword represents the bit exists and the other represents missing. By defining a Hamming loss between the predicted labels and true labels, the formulation could be derived into a set of Structured Support Vector Machine problems, which can be solved using efficient on-line softwares, namely *SSVM-MATLAB* [1]. It will be beneficial to contribute a new approach to the multi-label learning research field.

[1] `http://www.robots.ox.ac.uk/~vedaldi/svmstruct.html`

APPENDIX A: PROOF OF THEOREM 2

In order to solve Problem (3.3) and Problem (3.4), the following Proximal Subgradient Descent (PSD) update scheme is used:

$$\boldsymbol{w}_{t+\frac{1}{2}} = \boldsymbol{w}_t - \eta \boldsymbol{g}_t^f, \tag{A.1}$$

$$\boldsymbol{w}_{t+1} = arg\min_{w} \left\{ \frac{1}{2} \left\| \boldsymbol{w} - \boldsymbol{w}_{t+\frac{1}{2}} \right\|^2 + \eta r(\boldsymbol{w}) \right\}. \tag{A.2}$$

Above, $\boldsymbol{g}_t^f \in \partial f(\boldsymbol{w}_t)$ and η is a fixed step length. PSD first computes the unconstrained subgradient with respect to f.

In the second step, we find a new $\boldsymbol{w}_t$ from the intermediate result $\boldsymbol{w}_{t+\frac{1}{2}}$. By the first order optimality condition, with the minimizer $\boldsymbol{w}$, it holds that:

$$\mathbf{0} \in \partial \left\{ \frac{1}{2} \left\| \boldsymbol{w} - \boldsymbol{w}_{t+\frac{1}{2}} \right\|^2 + \eta r(\boldsymbol{w}) \right\} \Bigg|_{\boldsymbol{w}=\boldsymbol{w}_{t+1}}.$$

In light of Eq. (A.1), the above property amounts to:

$$\mathbf{0} \in \boldsymbol{w}_{t+1} - \boldsymbol{w}_t + \eta \boldsymbol{g}_t^f + \eta \partial r(\boldsymbol{w}_{t+1}). \tag{A.3}$$

Since $\boldsymbol{w}_{t+1}$ is the minimizer of Eq. (A.2), there is a vector $\boldsymbol{g}_{t+1}^r \in \partial r(\boldsymbol{w}_{t+1})$ such that Eq. (A.3) holds, *i.e.*

$$\mathbf{0} = \boldsymbol{w}_{t+1} - \boldsymbol{w}_t + \eta \boldsymbol{g}_t^f + \eta \boldsymbol{g}_{t+1}^r.$$

Finally, we have the following PSD update rule:

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta \boldsymbol{g}_t^f - \eta \boldsymbol{g}_{t+1}^r. \tag{A.4}$$

With the definitions of $\|\partial f(\boldsymbol{w})\|$ and $\|\partial r(\boldsymbol{w})\|$ in Section 3, we provide Lemma 3 as follows. Note that, unless specified otherwise, $\|\cdot\|$ will stand for the L_2 norm.

Lemma 3. *Assume that the subgradients of f and r are bounded as in Eq. (3.15) for some positive scalars A and G. Let $\eta \geq 0$ be a fixed step length and $\boldsymbol{w}^*$ be the minimizer of $f(\boldsymbol{w}) + r(\boldsymbol{w})$. Then, for a constant $c \leq 4$ we have:*

$$\begin{aligned} 2\eta(1 - cA\eta) f(\boldsymbol{w}_t) + 2\eta(1 - cA\eta) r(\boldsymbol{w}_{t+1}) \leq 2\eta f(\boldsymbol{w}^*) + 2\eta r(\boldsymbol{w}^*) + \|\boldsymbol{w}_t - \boldsymbol{w}^*\|^2 \\ - \|\boldsymbol{w}_{t+1} - \boldsymbol{w}^*\|^2 + 8\eta^2 G^2. \end{aligned} \tag{A.5}$$

Proof. By the definition of the subgradient, $\boldsymbol{g}_{t+1}^r \in \partial r(\boldsymbol{w} + \mathbf{1})$ and r's convexity:

$$\begin{aligned} r(\boldsymbol{w}^*) &\geq r(\boldsymbol{w}_{t+1}) + \left\langle \boldsymbol{g}_{t+1}^r, \boldsymbol{w}^* - \boldsymbol{w}_{t+1} \right\rangle \\ \Rightarrow \; - \left\langle \boldsymbol{g}_{t+1}^r, \boldsymbol{w}_{t+1} - \boldsymbol{w}^* \right\rangle &\leq r(\boldsymbol{w}^*) - r(\boldsymbol{w}_{t+1}), \end{aligned} \tag{A.6}$$

Additionally, the following relations hold:

$$
\begin{aligned}
\left\langle \boldsymbol{g}_{t+1}^r, \boldsymbol{w}_{t+1} - \boldsymbol{w}_t \right\rangle &= \left\langle \boldsymbol{g}_{t+1}^r, -\eta \boldsymbol{g}_t^f - \eta \boldsymbol{g}_{t+1}^r \right\rangle \leq \\
&\overset{Eq.\ (A.4)}{\leq} \left\| \boldsymbol{g}_{t+1}^r \right\| \left\| \eta \boldsymbol{g}_t^f + \eta \boldsymbol{g}_{t+1}^r \right\| \leq \\
&\leq \eta \left\| \boldsymbol{g}_{t+1}^r \right\|^2 + \eta \left\| \boldsymbol{g}_t^f \right\| \left\| \boldsymbol{g}_{t+1}^r \right\| \leq \\
&\overset{Eq.\ (3.15)}{\leq} \eta(Ar(\boldsymbol{w}_{t+1}) + G^2) + \eta(A\, max\{f(\boldsymbol{w}_t), r(\boldsymbol{w}_{t+1})\} + G^2), \qquad \text{(A.7)}
\end{aligned}
$$

where the second step is due to Cauchy-Schwarz inequality.

Now we relate $\|\boldsymbol{w}_{t+1} - \boldsymbol{w}^*\|$ to $\|\boldsymbol{w}_t - \boldsymbol{w}^*\|$ as follows:

$$
\begin{aligned}
&\|\boldsymbol{w}_{t+1} - \boldsymbol{w}^*\|^2 = \left\| \boldsymbol{w}_t - \eta \boldsymbol{g}_t^f - \eta \boldsymbol{g}_{t+1}^r - \boldsymbol{w}^* \right\|^2 = \\
&= \|\boldsymbol{w}_t - \boldsymbol{w}^*\|^2 - 2(\eta \left\langle \boldsymbol{g}_t^f, \boldsymbol{w}_t - \boldsymbol{w}^* \right\rangle + \eta \left\langle \boldsymbol{g}_{t+1}^r, \boldsymbol{w}_t - \boldsymbol{w}^* \right\rangle) + \left\| \eta \boldsymbol{g}_t^f + \eta \boldsymbol{g}_{t+1}^r \right\|^2 = \\
&= \|\boldsymbol{w}_t - \boldsymbol{w}^*\|^2 - 2\eta \left\langle \boldsymbol{g}_t^f, \boldsymbol{w}_t - \boldsymbol{w}^* \right\rangle + \eta^2 \left\| \boldsymbol{g}_t^f + \boldsymbol{g}_{t+1}^r \right\|^2 \\
&\quad - 2\eta(\left\langle \boldsymbol{g}_{t+1}^r, \boldsymbol{w}_{t+1} - \boldsymbol{w}^* \right\rangle - \left\langle \boldsymbol{g}_{t+1}^r, \boldsymbol{w}_{t+1} - \boldsymbol{w}_t \right\rangle). \qquad \text{(A.8)}
\end{aligned}
$$

In Eq. (A.8), $\eta^2 \left\| \boldsymbol{g}_t^f + \boldsymbol{g}_{t+1}^r \right\|^2$ can be bounded as follows:

$$
\begin{aligned}
&\eta^2 \left\| \boldsymbol{g}_t^f + \boldsymbol{g}_{t+1}^r \right\|^2 = \\
&= \eta^2 \left\| \boldsymbol{g}_t^f \right\|^2 + 2\eta^2 \left\langle \boldsymbol{g}_t^f, \boldsymbol{g}_{t+1}^r \right\rangle + \eta^2 \left\| \boldsymbol{g}_{t+1}^r \right\|^2 \leq \\
&\leq \eta^2 (Af(\boldsymbol{w}_t) + G^2) + 2\eta^2 A \, max\{f(\boldsymbol{w}_t), r(\boldsymbol{w}_{t+1})\} + \eta^2 (Ar(\boldsymbol{w}_{t+1}) + G^2) = \\
&= \eta^2 Af(\boldsymbol{w}_t) + 2\eta^2 A \, max\{f(\boldsymbol{w}_t), r(\boldsymbol{w}_{t+1})\} + \eta^2 Ar(\boldsymbol{w}_{t+1}) + 4\eta^2 G^2. \qquad \text{(A.9)}
\end{aligned}
$$

When Eq. (A.7) and Eq. (A.9) are substituted into Eq. (A.8), which obtain:

$$
\begin{aligned}
\|\boldsymbol{w}_{t+1} - \boldsymbol{w}^*\|^2 \leq \; & \|\boldsymbol{w}_t - \boldsymbol{w}^*\|^2 - 2\eta \left\langle \boldsymbol{g}_t^f, \boldsymbol{w}_t - \boldsymbol{w}^* \right\rangle - 2\eta \left\langle \boldsymbol{g}_{t+1}^r, \boldsymbol{w}_{t+1} - \boldsymbol{w}^* \right\rangle + \\
& + \eta^2 Af(\boldsymbol{w}_t) + 3\eta^2 Ar(\boldsymbol{w}_{t+1}) + \\
& + 4\eta^2 A \; \max\{f(\boldsymbol{w}_t), r(\boldsymbol{w}_{t+1})\} + 8\eta^2 G^2. \qquad \text{(A.10)}
\end{aligned}
$$

The convexities of both of $f(\boldsymbol{w})$ and $r(\boldsymbol{w})$ imply that:

$$
-\left\langle \boldsymbol{g}_t^f, \boldsymbol{w}_t - \boldsymbol{w}^* \right\rangle \leq f(\boldsymbol{w}^*) - f(\boldsymbol{w}_t), \qquad \text{(A.11)}
$$

$$
-\left\langle \boldsymbol{g}_{t+1}^r, \boldsymbol{w}_{t+1} - \boldsymbol{w}^* \right\rangle \leq r(\boldsymbol{w}^*) - r(\boldsymbol{w}_{t+1}). \qquad \text{(A.12)}
$$

The following also holds:

$$
\max\{f(\boldsymbol{w}_t), r(\boldsymbol{w}_{t+1})\} \leq f(\boldsymbol{w}_t) + r(\boldsymbol{w}_{t+1}). \qquad \text{(A.13)}
$$

By substituting Eq. (A.11), Eq. (A.12) and Eq. (A.13) into Eq. (A.10), we obtain

$$\begin{aligned}\|\boldsymbol{w}_{t+1}-\boldsymbol{w}^*\|^2 \leq & \|\boldsymbol{w}_t-\boldsymbol{w}^*\| + 8\eta^2G^2 + 2\eta[f(\boldsymbol{w}^*) - (1-\frac{5}{2}\eta A)f(\boldsymbol{w}_t)] \\ & + 2\eta[r(\boldsymbol{w}^*) - (1-\frac{7}{2}\eta A)r(\boldsymbol{w}_{t+1})] \leq \\ \leq & \|\boldsymbol{w}_t-\boldsymbol{w}^*\| + 8\eta^2G^2 + 2\eta[f(\boldsymbol{w}^*) - (1-c\eta A)f(\boldsymbol{w}_t)] \\ & + 2\eta[r(\boldsymbol{w}^*) - (1-c\eta A)r(\boldsymbol{w}_{t+1})]. \end{aligned} \tag{A.14}$$

By choosing $c \leq 4$, the second inequality holds. After some algebra, one can derive Eq. (A.5) from Eq. (A.14).

□

The following is the detailed proof of Theorem 2:

Proof. By Lemma 3, we have:

$$\begin{aligned}2\eta[(1-cA\eta)f(\boldsymbol{w}_T) - f(\boldsymbol{w}^*)] + & 2\eta[(1-cA\eta)r(\boldsymbol{w}_{t+1}) - r(\boldsymbol{w}^*)] \\ & \leq \|\boldsymbol{w}_t-\boldsymbol{w}^*\|^2 - \|\boldsymbol{w}_{t+1}-\boldsymbol{w}^*\|^2 + 8\eta^2G^2. \end{aligned} \tag{A.15}$$

Summing Eq. (A.15) over $t = 1, \ldots, T$ we get

$$\sum_{t=1}^{T} 2\eta[(1-cA\eta)f(\boldsymbol{w}_T) - f(\boldsymbol{w}^*)] + 2\eta[(1-cA\eta)r(\boldsymbol{w}_{t+1}) - r(\boldsymbol{w}^*)] \leq$$

$$\leq \|\boldsymbol{w}_1 - \boldsymbol{w}^*\|^2 - \|\boldsymbol{w}_{T+1} - \boldsymbol{w}^*\|^2 + 8T\eta^2 G^2 \leq$$

$$\leq \|\boldsymbol{w}_1 - \boldsymbol{w}^*\|^2 + 8T\eta^2 G^2 \leq$$

$$\leq D^2 + 8T\eta^2 G^2. \tag{A.16}$$

The last inequality holds because $\|\boldsymbol{w}^*\|^2 \leq D$ and $\boldsymbol{w}_1 = 0$ as described in Theorem 2. For part of Eq. (A.16), it holds:

$$\begin{aligned}
&\sum_{t=1}^{T} \eta[(1-cA\eta)r(\boldsymbol{w}_{t+1}) - r(\boldsymbol{w}^*)] = \\
=&\sum_{t=1}^{T} \eta[(1-cA\eta)r(\boldsymbol{w}_t) - \boldsymbol{w}^*] - \eta[(1-cA\eta)r(\boldsymbol{w}_1) - r(\boldsymbol{w}^*)] \\
&+ \eta[(1-cA\eta)r(\boldsymbol{w}_{T+1}) - r(\boldsymbol{w}^*)] = \\
=&\sum_{t=1}^{T} \eta[(1-cA\eta)r(\boldsymbol{w}_t) - \boldsymbol{w}^*] + \eta(1-cA\eta)r(\boldsymbol{w}_{T+1}) \geq \\
\geq&\sum_{t=1}^{T} \eta[(1-cA\eta)r(\boldsymbol{w}_t) - \boldsymbol{w}^*].
\end{aligned} \tag{A.17}$$

The second equality holds due to the assumptions that $\boldsymbol{w}_1 = \mathbf{0}$ and $r(\mathbf{0}) = 0$. Besides, given the step length η, this term $\eta(1-cA\eta)r(\boldsymbol{w}_{T+1})$ is larger than 0, which establishes the last inequality. Now, when substituting Eq. (A.17) back into Eq. (A.16), we get

$$\sum_{t=1}^{T} 2\eta[(1-cA\eta)r(\boldsymbol{w}_t) - \boldsymbol{w}^*] + 2\eta[(1-cA\eta)r(\boldsymbol{w}_t) - r(\boldsymbol{w}^*)] \leq D^2 + 8T\eta^2G^2$$

$$\Rightarrow 2\eta(1-cA\eta)\sum_{t=1}^{T}[f(\boldsymbol{w}_t) + r(\boldsymbol{w}_t)] - 2\eta T(f(\boldsymbol{w}^*) + r(\boldsymbol{w}^*)) \leq D^2 + 8T\eta^2G^2$$

$$\Rightarrow \sum_{t=1}^{T} f(\boldsymbol{w}_t) + r(\boldsymbol{w}_t) \leq \frac{8T\eta^2G^2}{2\eta(1-cA\eta)} + \frac{T(f(\boldsymbol{w}^*) + r(\boldsymbol{w}^*))}{1-cA\eta}. \tag{A.18}$$

Additionally, the following holds:

$$\underset{t\in\{1\ldots T\}}{min} f(\boldsymbol{w}_t) + r(\boldsymbol{w}_t) \leq \frac{1}{T}\sum_{t=1}^{T} f(\boldsymbol{w}_t) + r(\boldsymbol{w}_t). \tag{A.19}$$

Based on Eq. (A.18), Eq. (A.19) and choosing $\eta = \frac{D}{\sqrt{8T}G}$, we obtain the main result shown by Theorem 2.

□

APPENDIX B: PROOF OF THEOREM 3

Here, we provide the detailed proof of Theorem 3 in subsection **Analysis** in Section 3.

Proof. Firstly, we need to prove that each of the two or three block minimizations in our algorithms decrease the objective function value under consideration. For the first block minimization, this is true, implied by Theorem 2. For the second block, beausee of the Majorization Minimization (MM) algorithm, we have the following inequalities:

$$q(\boldsymbol{g}^*) = q(\boldsymbol{g}^*|\boldsymbol{g}^*) \leq q(\boldsymbol{g}^*|\boldsymbol{g}') \leq q(\boldsymbol{g}'|\boldsymbol{g}') = q(\boldsymbol{g}'). \tag{B.1}$$

Eq. (B.1) implies that the second block minimization does not increase the objective function value. The optimal algorithm for the third block also guarantees the non-increasing nature of the cost function. Additionally, note that the objective function is lower-bounded. Therefore, Algorithm 1 converges.

In the next step, we prove that the set of fixed points of the Algorithm 1 includes the Karush-Kuhn-Tucker (KKT) points of Problem (3.4). Suppose the algorithm converges to a KKT point $\left\{\boldsymbol{L}^{k*}, \boldsymbol{g}^{k*}\right\}_{k\in\mathbb{N}_K}$; then, we prove that this point is also a fixed point of the algorithm's iterative map. Denote $f_1(\boldsymbol{g}^k)$, $h_1(\boldsymbol{g}^k)$ and $f_0(\boldsymbol{L}^k, \boldsymbol{g}^k)$ as the inequality constraint, equality constraint and the cost function of Problem (3.4). A KKT point will satisfy the following by definition:

$$\begin{aligned} \mathbf{0} \in\ & \partial_{\boldsymbol{L}^k} f_0(\boldsymbol{L}^{k*}, \boldsymbol{g}^{k*}) + \bigtriangledown_{\boldsymbol{g}^k} f_0(\boldsymbol{L}^{k*}, \boldsymbol{g}^{k*}) \\ & - (\boldsymbol{\beta}^k)^T \bigtriangledown_{\boldsymbol{g}^k} f_1(\boldsymbol{g}^{k*}) + \boldsymbol{\alpha}^T \bigtriangledown_{\boldsymbol{g}^k} h_1(\boldsymbol{g}^{k*}), \quad k \in \mathbb{N}_K. \end{aligned} \tag{B.2}$$

In relation to Problem (3.7), which is solved by the second block, by setting the gradient of the

problem's Lagrangian to $\mathbf{0}$, the KKT point satisfies the following equality:

$$2\tilde{\boldsymbol{S}}\boldsymbol{g}^* - \boldsymbol{\beta} - \boldsymbol{B}^T\boldsymbol{\alpha} = \mathbf{0}. \tag{B.3}$$

Problem (3.8) can be solved by Theorem 1; We obtain that

$$\boldsymbol{g} = -\frac{1}{2\boldsymbol{\mu}}(\boldsymbol{B}^T\boldsymbol{\alpha} + \boldsymbol{\beta} - 2\boldsymbol{H}\boldsymbol{g}^*). \tag{B.4}$$

Substituting Eq. (B.3) and $\boldsymbol{H} = \tilde{\boldsymbol{S}} + \mu\boldsymbol{I}$ into Eq. (B.4), we obtain

$$\boldsymbol{g} = -\frac{1}{2\boldsymbol{\mu}}(\boldsymbol{B}^T\boldsymbol{\alpha} + \boldsymbol{\beta} - 2\boldsymbol{H}\boldsymbol{g}^*) = -\frac{1}{2\boldsymbol{\mu}}(2\tilde{\boldsymbol{S}}\boldsymbol{g}^* - 2\tilde{\boldsymbol{S}}\boldsymbol{g}^* - 2\boldsymbol{\mu}\boldsymbol{g}^*) = \boldsymbol{g}^*. \tag{B.5}$$

The aforementioned equation indicates that the second block of Algorithm 1 does not update the solution. If Eq. (B.2) is substituted by Eq. (B.3), we have $\mathbf{0} \in \partial_{\boldsymbol{L}^k} f_0(\boldsymbol{L}^{k*}, \boldsymbol{g}^{k*})$ for all k. This is the optimality condition for the subgradient algorithm; the first block does not update the solution either. Therefore, a Problem (3.4)'s KKT point is a fixed point of Algorithm 1.

Finally, we prove that, for Problem (3.3), the KKT points are also included in the set of fixed points of Algorithm 1. Assume Algorithm 1 has converged to a KKT point $\left\{\boldsymbol{L}^{k*}, \boldsymbol{g}^{k*}\right\}_{k\in\mathbb{N}_K}$ and S^* is the true similarity matrix. Similar to the previous proof, we start from the second block. Following the same procedure, we find the second block will not update the solution of vector $\boldsymbol{g}^*$. Now, during the third block minimization, the ψ_{mn} quantities remain unchanged, since $\boldsymbol{g}^*$ does not change. The minimization procedure we proposed for the third block will leave the similarity matrix unchanged, since the coefficient matrix with elements ψ_{mn} is fixed. Now, if Eq. (B.3) is substituted back into

Eq. (B.2), we obtain the optimality condition of the first block optimization. Thus, the solution is not updated in the first block. Thus, a Problem (3.3)'s KKT point is a fixed point of Algorithm 1.

□

APPENDIX C: PROOF OF PROPOSITION 3

Proof. By replacing hinge function in Problem (4.6), for the first block minimization, we got the following problem:

$$\begin{aligned}
\underset{\substack{w_{b,m},\beta_b \\ \xi^b_{c,n,s}}}{min} \quad & \lambda_1 \sum_c \sum_s \sum_n \gamma'_{c,n,s} \xi^b_{c,n,s} + \frac{1}{2} \sum_m \frac{\|w_{b,m}\|^2_{\mathcal{H}_m}}{\theta_{b,m}} \\
\text{s.t.} \quad & \xi^b_{c,n,s} \geq 0 \\
& \xi^b_{c,n,s} \geq 1 - (\sum_m \langle w_{b,m}, \phi_m(x) \rangle_{\mathcal{H}_m} + \beta_b) \mu^b_{c,s}
\end{aligned} \tag{C.1}$$

First of all, after considering Representer Theorem [97], we have:

$$w_{b,m} = \theta_{b,m} \sum_n \eta_{b,n} \phi_m(x_n) \tag{C.2}$$

Here, n is defined as the training samples' index. By defining $\boldsymbol{\xi}_b \in \mathbb{R}^{NCS}$ to be the vecotr containing all $\xi^b_{c,n,s}$'s, $\boldsymbol{\eta}_b \triangleq [\eta_{b,1}, \eta_{b,2}, ..., \eta_{b,N}]^T \in \mathbb{R}^N$ and $\boldsymbol{\mu}_b \triangleq [\mu^b_{1,1}, \ldots, \mu^b_{1,S}, \mu^b_{2,1}, \ldots, \mu^b_{C,S}]^T \in \mathbb{R}^{CS}$, the vectorized version of Problem (C.1) with Eq. (C.2):

$$\begin{aligned}
\underset{\boldsymbol{\eta}_b, \boldsymbol{\xi}_b, \beta_b}{min} \quad & \lambda_1 \boldsymbol{\gamma}' \boldsymbol{\xi}_b + \frac{1}{2} \boldsymbol{\eta}_b^T \mathbf{K}_b \boldsymbol{\eta}_b \\
s.t. \quad & \boldsymbol{\xi}_b \succeq \mathbf{0} \\
& \boldsymbol{\xi}_b \succeq \mathbf{1}_{NCS} - (\boldsymbol{\mu}_b \otimes \mathbf{K}_b) \boldsymbol{\eta}_b - (\boldsymbol{\mu}_b \otimes \mathbf{1}_N) \beta_b
\end{aligned} \tag{C.3}$$

Here, Prop. 3 has already defined $\boldsymbol{\gamma}'$ and $\mathbf{K}_b$. Take the Lagrangian $\mathcal{L}$ and its derivatives, we have the following relations, here $\boldsymbol{\alpha}_b$ and $\boldsymbol{\zeta}_b$ are Lagrangian multipliers for the two constraints:

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\xi}_b} = \mathbf{0} \Rightarrow \begin{cases} \boldsymbol{\zeta}_b = \lambda_1 \boldsymbol{\gamma}' - \boldsymbol{\alpha}_b \\ \mathbf{0} \preceq \boldsymbol{\alpha}_b \preceq \lambda_1 \boldsymbol{\gamma}' \end{cases} \tag{C.4}$$

$$\frac{\partial \mathcal{L}}{\partial \beta_b} = 0 \Rightarrow \boldsymbol{\alpha}_b^T (\boldsymbol{\mu}_b \otimes \mathbf{1}_N) = 0 \tag{C.5}$$

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\eta}_b} = \mathbf{0} \overset{\exists \mathbf{K}_b^{-1}}{\Rightarrow} \boldsymbol{\eta}_b = \mathbf{K}_b^{-1} (\boldsymbol{\mu}_b \otimes \mathbf{K}_b)^T \boldsymbol{\alpha}_b \tag{C.6}$$

Substitute Eq. (C.4), Eq. (C.5) and Eq. (C.6) back into $\mathcal{L}$, meanwhile, we notice the quatric term becomes:

$$\begin{aligned} &(\boldsymbol{\mu}_b \otimes \mathbf{K}_b)\mathbf{K}_b^{-1}(\boldsymbol{\mu}_b^T \otimes \mathbf{K}_b) = \\ =&(\boldsymbol{\mu}_b \otimes \mathbf{K}_b)(1 \otimes \mathbf{K}_b^{-1})(\boldsymbol{\mu}_b^T \otimes \mathbf{K}_b) = \\ =&(\boldsymbol{\mu}_b \otimes \mathbf{I}_{N\times N})(\boldsymbol{\mu}_b^T \otimes \mathbf{K}_b) = \\ =&(\boldsymbol{\mu}_b \boldsymbol{\mu}_b^T) \otimes \mathbf{K}_b \end{aligned} \tag{C.7}$$

Eq. (C.7) can be further derived:

$$
\begin{aligned}
&(\boldsymbol{\mu}_b \boldsymbol{\mu}_b^T) \otimes \mathbf{K}_b = \\
&= [(\mathrm{diag}\,(\boldsymbol{\mu}_b)\,\mathbf{1}_C)(\mathrm{diag}\,(\boldsymbol{\mu}_b)\,\mathbf{1}_C)^T] \otimes \mathbf{K}_b = \\
&= [\mathrm{diag}\,(\boldsymbol{\mu}_b)\,(\mathbf{1}_C \mathbf{1}_C^T)\,\mathrm{diag}\,(\boldsymbol{\mu}_b)] \otimes [\mathbf{I}_N \mathbf{K}_b \mathbf{I}_N] = \\
&= [\mathrm{diag}\,(\boldsymbol{\mu}_b) \otimes \mathbf{I}_N][(\mathbf{1}_C \mathbf{1}_C^T) \otimes \mathbf{K}_b][\mathrm{diag}\,(\boldsymbol{\mu}_b) \otimes \mathbf{I}_N] = \\
&= [\mathrm{diag}\,(\boldsymbol{\mu}_b \otimes \mathbf{1}_N)][(\mathbf{1}_C \mathbf{1}_C^T) \otimes \mathbf{K}_b][\mathrm{diag}\,(\boldsymbol{\mu}_b \otimes \mathbf{1}_N)] = \\
&= \mathbf{D}_b[(\mathbf{1}_C \mathbf{1}_C^T) \otimes \mathbf{K}_b]\mathbf{D}_b
\end{aligned}
\tag{C.8}
$$

The first equality comes from $\mathrm{diag}\,(\boldsymbol{u})\,\mathbf{1} = \boldsymbol{u}$ for some vector $\boldsymbol{u}$. The mixed-product property of Kronecker product derives the third equality. This relation $\mathrm{diag}\,(\boldsymbol{u} \otimes \mathbf{1}) = \mathrm{diag}\,(\boldsymbol{u}) \otimes \mathbf{I}$ gives the fourth equality. $\mathbf{D}_b$ is defined in Prop. 3.

After considering Eq. (C.7) and Eq. (C.8), we get the final dual form as shown in Prop. 3. □

APPENDIX D: PROOF OF PROPOSITION 4

Proof. With the definitions of proximal operator Eq. (4.11), we have the following problem to minimize over:

$$
\begin{aligned}
P(\boldsymbol{\mu}) &= \eta \left\| U\boldsymbol{\mu} \right\|_2 + \frac{1}{2} \left\| \boldsymbol{v} - \boldsymbol{\mu} \right\|_2^2 = \\
&= \eta \left\| U\boldsymbol{\mu} \right\|_2 + \frac{1}{2} \left\| \boldsymbol{v}_1 - \boldsymbol{\mu}_1 \right\|_2^2 + \cdots + \frac{1}{2} \left\| \boldsymbol{v}_S - \boldsymbol{\mu}_S \right\|_2^2
\end{aligned} \tag{D.1}
$$

The definitions of the vectors $\boldsymbol{\mu}$ and $\boldsymbol{v}$ generate the second equality. Since L_2 norm is non differentiable at point 0, we optimize Eq. (D.1) in two cases.

Case 1: when $\boldsymbol{\mu}_i \neq \boldsymbol{\mu}_j$, we take the gradients for each individual $\boldsymbol{\mu}_1$ to $\boldsymbol{\mu}_S$:

$$
\begin{cases}
\frac{\partial P(\boldsymbol{\mu})}{\partial \boldsymbol{\mu}_1} = \boldsymbol{\mu}_1 - \boldsymbol{v}_1 = \mathbf{0} \\
\qquad \vdots \\
\frac{\partial P(\boldsymbol{\mu})}{\partial \boldsymbol{\mu}_i} = \eta \frac{\boldsymbol{\mu}_i - \boldsymbol{\mu}_j}{\left\| \boldsymbol{\mu}_i - \boldsymbol{\mu}_j \right\|_2} + \boldsymbol{\mu}_i - \boldsymbol{v}_i = \mathbf{0} \\
\qquad \vdots \\
\frac{\partial P(\boldsymbol{\mu})}{\partial \boldsymbol{\mu}_j} = \eta \frac{\boldsymbol{\mu}_j - \boldsymbol{\mu}_i}{\left\| \boldsymbol{\mu}_i - \boldsymbol{\mu}_j \right\|_2} + \boldsymbol{\mu}_j - \boldsymbol{v}_j = \mathbf{0} \\
\qquad \vdots \\
\frac{\partial P(\boldsymbol{\mu})}{\partial \boldsymbol{\mu}_S} = \boldsymbol{\mu}_S - \boldsymbol{v}_S = \mathbf{0}
\end{cases} \tag{D.2}
$$

Solve the linear equations with $\boldsymbol{\mu}_i$ and $\boldsymbol{\mu}_j$:

$$\begin{cases} \boldsymbol{\mu}_i = \frac{1}{2\tau+1}\left[(1+\tau)\boldsymbol{v}_i + \tau\boldsymbol{v}_j\right] \\ \boldsymbol{\mu}_j = \frac{1}{2\tau+1}\left[\tau\boldsymbol{v}_i + (1+\tau)\boldsymbol{v}_j\right] \end{cases} \tag{D.3}$$

where $\tau \triangleq \eta/\delta$ and $\delta \triangleq \left\|\boldsymbol{\mu}_i - \boldsymbol{\mu}_j\right\|_2$. Now we have the following derivations:

$$\begin{aligned} &\boldsymbol{\mu}_i - \boldsymbol{\mu}_j = \frac{1}{2\tau+1}(\boldsymbol{v}_i - \boldsymbol{v}_j) \\ &\Rightarrow \left\|\boldsymbol{\mu}_i - \boldsymbol{\mu}_j\right\|_2 = \frac{1}{2\tau+1}\left\|\boldsymbol{v}_i - \boldsymbol{v}_j\right\|_2 \\ &\Rightarrow \delta = \frac{\left\|\boldsymbol{v}_i - \boldsymbol{v}_j\right\|_2}{2\frac{\eta}{\delta}+1} \\ &\Rightarrow \delta = \left\|\boldsymbol{\mu}_i - \boldsymbol{\mu}_j\right\|_2 = \left\|\boldsymbol{v}_i - \boldsymbol{v}_j\right\|_2 - 2\eta \end{aligned} \tag{D.4}$$

Plug τ and Eq. (D.4) into Eq. (D.3), we achieve the results for $\boldsymbol{\mu}_i$ and $\boldsymbol{\mu}_j$:

$$\begin{cases} \boldsymbol{\mu}_i = \alpha_1\boldsymbol{v}_i + \alpha_2\boldsymbol{v}_j \\ \boldsymbol{\mu}_j = \alpha_2\boldsymbol{v}_i + \alpha_1\boldsymbol{v}_j \end{cases} \tag{D.5}$$

Here $\alpha_1 = 1 - \alpha_2$ and $\alpha_2 = \frac{\eta}{\|\boldsymbol{v}_i - \boldsymbol{v}_j\|_2}$. Additionally, Eq. (D.4) is larger than 0 which gives the following condition for **Case 1**: $0 < \eta \leq \frac{\|\boldsymbol{v}_i - \boldsymbol{v}_j\|_2}{2}$.

Case 2: when $\boldsymbol{\mu}_i = \boldsymbol{\mu}_j$, $\boldsymbol{\mu}_i$ and $\boldsymbol{\mu}_j$ are represented as:

$$\begin{cases} \boldsymbol{\mu}_i = \frac{1}{2}\boldsymbol{v}_i + \frac{1}{2}\boldsymbol{v}_j \\ \boldsymbol{\mu}_j = \frac{1}{2}\boldsymbol{v}_i + \frac{1}{2}\boldsymbol{v}_j \end{cases} \tag{D.6}$$

Note that this case satisfies when $\eta > \frac{\|\boldsymbol{v}_i - \boldsymbol{v}_j\|_2}{2}$.

Combining two cases, the results provided in Prop. 4 are achieved.

□

APPENDIX E: PROOF OF LEMMA 1

Proof. From Definition 3, we have:

$$\begin{aligned}
\widehat{\Re}_Q\left(\Psi \circ \widetilde{\mathcal{F}}\right) &= \frac{1}{N}\,\mathbb{E}_\sigma\left\{\sup_{\boldsymbol{f}\in\widetilde{\boldsymbol{F}}}\sum_n \sigma_n \Psi(\boldsymbol{f}(z_n))\right\} = \\
&= \frac{1}{N}\,\mathbb{E}_{\boldsymbol{\sigma}_{N-1}}\left\{\mathbb{E}_{\sigma_N}\left\{\sup_{\boldsymbol{f}\in\widetilde{\mathcal{F}}}\left[u(\boldsymbol{f}) + \sigma_N\Psi(\boldsymbol{f}(z_N))\right]\right\}\right\} \\
&= \frac{1}{N}\,\mathbb{E}_{\boldsymbol{\sigma}_{N-1}}\left\{A(\boldsymbol{\sigma}_{N-1})\right\} \qquad \text{(E.1)}
\end{aligned}$$

where $u(\boldsymbol{f}) \triangleq \sum_{n=1}^{N-1}\sigma_n\Psi(\boldsymbol{f}(z_n))$ and $A(\boldsymbol{\sigma}_{N-1}) \triangleq \mathbb{E}_{\sigma_N}\left\{\sup_{\boldsymbol{f}\in\widetilde{\mathcal{F}}}\left[u(\boldsymbol{f}) + \sigma_N\Psi(\boldsymbol{f}(z_N))\right]\right\}$.

Expanding the expectation , we get:

$$A(\boldsymbol{\sigma}_{N-1}) = \frac{1}{2}[\sup_{\boldsymbol{f}\in\widetilde{\mathcal{F}}}\left[u(\boldsymbol{f}) + \Psi(\boldsymbol{f}(z_n))\right] + \sup_{\boldsymbol{f}\in\widetilde{\mathcal{F}}}\left[u(\boldsymbol{f}) - \Psi(\boldsymbol{f}(z_n))\right]] \qquad \text{(E.2)}$$

Additionally, we define the following: $\widehat{B}(\boldsymbol{f}) \triangleq u(\boldsymbol{f}) + \Psi(\boldsymbol{f}(z_n))$ and $\widetilde{B}(\boldsymbol{f}) \triangleq u(\boldsymbol{f}) - \Psi(\boldsymbol{f}(z_n))$.

From the superium's definition, we have that $\forall\epsilon > 0$, there are $\widehat{\boldsymbol{f}}$ amd $\widetilde{\boldsymbol{f}}$ in $\widetilde{\mathcal{F}}$ such that:

$$\sup_{\boldsymbol{f}\in\widetilde{\mathcal{F}}}\widehat{B}(\boldsymbol{f}) \geq \widehat{B}(\widehat{\boldsymbol{f}}) \geq (1-\epsilon)\sup_{\boldsymbol{f}\in\widetilde{\mathcal{F}}}\widehat{B}(\boldsymbol{f}) \qquad \text{(E.3)}$$

$$\sup_{\boldsymbol{f}\in\widetilde{\mathcal{F}}}\widetilde{B}(\boldsymbol{f}) \geq \widetilde{B}(\widehat{\boldsymbol{f}}) \geq (1-\epsilon)\sup_{\boldsymbol{f}\in\widetilde{\mathcal{F}}}\widetilde{B}(\boldsymbol{f}) \qquad \text{(E.4)}$$

From Eq. (E.3) and Eq. (E.4), for any $\epsilon > 0$, we have:

$$(1-\epsilon)A(\boldsymbol{\sigma}_{N-1}) \leq \frac{1}{2}\left[\widehat{B}(\widehat{\boldsymbol{f}}) + \widetilde{B}(\widetilde{\boldsymbol{f}})\right] = \frac{1}{2}\left[u(\widehat{\boldsymbol{f}}) + u(\widetilde{\boldsymbol{f}}) + \Psi(\widehat{\boldsymbol{f}}(z_n)) - \Psi(\widetilde{\boldsymbol{f}}(z_n))\right] \tag{E.5}$$

Since Ψ is L-Lipschitz continuous w.r.t the $\|\cdot\|_1$ norm, it holds that:

$$\begin{aligned}
\Psi(\widehat{\boldsymbol{f}}(z_n)) - \Psi(\widetilde{\boldsymbol{f}}(z_n)) &\leq L\left\|\widehat{\boldsymbol{f}}(z_n) - \widetilde{\boldsymbol{f}}(z_n)\right\|_1 = \\
&= L\sum_{b=1}^{B}|\widehat{f}_b(z_n) - \widetilde{f}_b(z_n)| = \\
&= L\sum_{b=1}^{B} q_b\left(\widehat{f}_b(z_n) - \widetilde{f}_b(z_n)\right)
\end{aligned} \tag{E.6}$$

where $q_b \triangleq sgn\left(\widehat{f}_b(z_n) - \widetilde{f}_b(z_n)\right)$. From Eq. (E.5) and Eq. (E.6), we obtain:

$$\begin{aligned}
&(1-\epsilon)A(\boldsymbol{\sigma}_{N-1}) \\
&\leq \frac{1}{2}\left[u(\widehat{\boldsymbol{f}}) + u(\widetilde{\boldsymbol{f}}) + L\sum_{b=1}^{B} q_b\left(\widehat{f}_b(z_n) - \widetilde{f}_b(z_n)\right)\right] = \\
&= \frac{1}{2}\left[\left[u(\widehat{\boldsymbol{f}}) + L\sum_{b=1}^{B} q_b\widehat{f}_b(z_N)\right] + \left[u(\widetilde{\boldsymbol{f}}) - L\sum_{b=1}^{B} q_b\widetilde{f}_b(z_N)\right]\right]
\end{aligned} \tag{E.7}$$

By the definition of superium, Eq. (E.7) is bounded by:

$$(E.7) \leq \sup_{\boldsymbol{q}\in\mathbb{H}^B} \frac{1}{2}\left[\left[u(\widehat{\boldsymbol{f}}) + L\sum_b^B q_b \widehat{f}_b(z_N)\right] + \left[u(\widetilde{\boldsymbol{f}}) - L\sum_b^B q_b \widetilde{f}(z_N)\right]\right] \leq$$

$$\leq \frac{1}{2}\left[\sup_{\boldsymbol{q}\in\mathbb{H}^B}\left[u(\widehat{\boldsymbol{f}}) + L\sum_b^B q_b \widehat{f}(z_N)\right]\right] + \sup_{\boldsymbol{q}\in\mathbb{H}^B}\left[u(\widetilde{\boldsymbol{f}}) - L\sum_b^B q_b \widetilde{f}_b(z_N)\right]\right] \tag{E.8}$$

With the help of Eq. (E.3) and Eq. (E.4), Eq. (E.8) is bounded:

$$(E.8) \leq \frac{1}{2}\left[\sup_{\boldsymbol{q}\in\mathbb{H}^B}\sup_{\boldsymbol{f}\in\widetilde{\mathcal{F}}}\left[u(\boldsymbol{f}) + L\sum_b^B q_b f_b(z_N)\right] + \sup_{\boldsymbol{q}\in\mathbb{H}^B}\sup_{\boldsymbol{f}\in\widetilde{\mathcal{F}}}\left[u(\boldsymbol{f}) - L\sum_b^B q_b f_b(z_N)\right]\right] =$$

$$= \mathbb{E}_{\sigma_N}\left\{\sup_{\boldsymbol{q}\in\mathbb{H}^B}\sup_{\boldsymbol{f}\in\widetilde{\mathcal{F}}}\left[u(\boldsymbol{f}) + \sigma_N L\sum_b^B q_b f_b(z_N)\right]\right\} =$$

$$= \mathbb{E}_{\sigma_N}\left\{\sup_{\boldsymbol{f}\in\widetilde{\mathcal{F}}}\left[u(\boldsymbol{f}) + \sigma_N L\sup_{\boldsymbol{q}\in\mathbb{H}^B}\sum_b^B q_b f_b(z_N)\right]\right\} =$$

$$= \mathbb{E}_{\sigma_N}\left\{\sup_{\boldsymbol{f}\in\widetilde{\mathcal{F}}}\left[u(\boldsymbol{f}) + \sigma_N L\sum_b^B \sup_{q_b\in\{\pm 1\}} q_b f_b(z_N)\right]\right\} =$$

$$= \mathbb{E}_{\sigma_N}\left\{\sup_{\boldsymbol{f}\in\widetilde{\mathcal{F}}}\left[u(\boldsymbol{f}) + \sigma_N L\sum_b^B sgn(f_b(z_N)) f_b(z_N)\right]\right\} =$$

$$= \mathbb{E}_{\sigma_N}\left\{\sup_{\boldsymbol{f}\in\widetilde{\mathcal{F}}}\left[u(\boldsymbol{f}) + \sigma_N L\left\|\boldsymbol{f}(z_N)\right\|_1\right]\right\} \tag{E.9}$$

Since Eq. (E.9) holds for every $\epsilon > 0$, we have that :

$$A(\boldsymbol{\sigma}_{N-1}) \leq \mathbb{E}_{\sigma_N}\left\{\sup_{\boldsymbol{f}\in\widetilde{\mathcal{F}}}\left[u(\boldsymbol{f}) + \sigma_N L\left\|\boldsymbol{f}(z_N)\right\|_1\right]\right\} \tag{E.10}$$

Repeating this process for the remaining σ eventually yields the result of this lemma. $\square$

APPENDIX F: PROOF OF LEMMA 2

Proof. Let $\Psi(\cdot) \triangleq \|\cdot\|_1$. Utilizing the similar technique in Lemma 1, by defining $u(\boldsymbol{f}) \triangleq \sum_{n=1}^{N-1} \sigma_N \Psi(\boldsymbol{f}(z_N))$, we have:

$$\widehat{\Re}_Q(\left\|\widetilde{\mathcal{F}}\right\|_1) = \frac{1}{N} \mathbb{E}_{\boldsymbol{\sigma}_{N-1}} \left\{A(\boldsymbol{\sigma}_{N-1})\right\} \tag{F.1}$$

Here $A(\boldsymbol{\sigma}_{N-1}) = \mathbb{E}_{\sigma_N} \left\{\sup_{\boldsymbol{f} \in \widetilde{\mathcal{F}}} \left[u(\boldsymbol{f}) + \sigma_N \Psi(\boldsymbol{f}(z_N))\right]\right\}$. Similarly, by defining $\widehat{B}(\boldsymbol{f}) \triangleq u(\boldsymbol{f}) + \Psi(\boldsymbol{f}(z_N))$ and $\widetilde{B}(\boldsymbol{f}) \triangleq u(\boldsymbol{f}) - \Psi(\boldsymbol{f}(z_N))$, we have for any $\epsilon > 0$:

$$\begin{aligned}(1-\epsilon)A(\boldsymbol{\sigma}_{N-1}) &\leq \frac{1}{2}\left[\widehat{B}(\widehat{\boldsymbol{f}}) + \widetilde{B}(\widetilde{\boldsymbol{f}})\right] = \\ &= \frac{1}{2}\left[u(\widehat{\boldsymbol{f}}) + u(\widetilde{\boldsymbol{f}}) + \Psi(\widehat{\boldsymbol{f}}(z_N)) - \Psi(\widetilde{\boldsymbol{f}}(z_N))\right]\end{aligned} \tag{F.2}$$

By the reverse triangle inequality and $|\cdot|$'s 1 - Lipschitz property:

$$\begin{aligned}\Psi(\widehat{\boldsymbol{f}}(z_N)) - \Psi(\widetilde{\boldsymbol{f}}(z_N)) &= \sum_{b=1}^{B}\left[|\widehat{f}_b(z_N)| - |\widetilde{f}_b(z_N)|\right] \leq \\ &\leq \sum_{b=1}^{B} sgn(\widehat{f}_b(z_N) - \widetilde{f}_b(z_N))(\widehat{f}_b(z_N) - \widetilde{f}_b(z_N))\end{aligned} \tag{F.3}$$

With the definition of $q_b \triangleq sgn(\widehat{f}_b(z_N) - \widetilde{f}_b(z_N))$, we combine Eq. (F.2) and Eq. (F.3):

$$
\begin{aligned}
&(1-\epsilon)A(\boldsymbol{\sigma}_{N-1}) \\
&\leq \frac{1}{2}\left[u(\widehat{\boldsymbol{f}}) + u(\widetilde{\boldsymbol{f}}) + \sum_{b=1}^{B} q_b(\widehat{f}_b(z_N) - \widetilde{f}_b(z_N))\right] = \\
&= \frac{1}{2}\left[\left[u(\widehat{\boldsymbol{f}}) + \sum_{b=1}^{B} q_b \widehat{f}_b(z_N)\right] + \left[u(\widetilde{\boldsymbol{f}}) - \sum_{b=1}^{B} q_b \widetilde{f}_b(z_N)\right]\right]
\end{aligned} \tag{F.4}
$$

For $b \in \mathbb{H}_B$, define $q'_b \triangleq q_b$ if $q_b \neq 0$ and $q'_b \triangleq 1$ otherwise. Also, define $\widehat{f}'(\cdot) \triangleq q'_b \widehat{f}(\cdot)$, then we have $\widehat{f}(\cdot) = q'_b \widehat{f}'(\cdot)$ and :

$$
\begin{aligned}
&u(\widehat{\boldsymbol{f}}) + \sum_{b=1}^{B} q_b \widehat{f}_b(z_N) = \\
&= u\left(q'_1 \widehat{f}'_1(\cdot), ..., q'_B \widehat{f}'_B(\cdot)\right) + \sum_{b=1}^{B} \widehat{f}'_b(z_N) = \\
&= \sum_{n=1}^{N-1} \sigma_n \sum_{b=1}^{B} |q'_b \widehat{f}'_b(z_n)| + \sum_{b=1}^{B} \widehat{f}'_b(z_N) = \\
&= u(\widehat{\boldsymbol{f}}') + \sum_{b=1}^{B} \widehat{f}'_b(z_N) \leq \sup_{\boldsymbol{f} \in \widetilde{\mathcal{F}}} \left[u(\boldsymbol{f}) + \sum_{b=1}^{B} f_b(z_N)\right]
\end{aligned} \tag{F.5}
$$

The above derivation is based on the fact that if $[f_1(z), ..., f_B(z)]^T \in \widetilde{\mathcal{F}}$, then we also have $[\pm f_1(z), ..., f_B(z)]^T \in \widetilde{\mathcal{F}}$ for $z \in \mathcal{Z}$.

Using a similar rationale, we can show that:

$$u(\widetilde{\boldsymbol{f}}) - \sum_{b=1}^{B} q_b \widetilde{f}_b(z_n) \leq \sup_{\boldsymbol{f} \in \widetilde{\mathcal{F}}} \left[u(\boldsymbol{f}) - \sum_{b=1}^{B} f_b(z_N) \right] \tag{F.6}$$

Combine Eq. (F.4), Eq. (F.5) and Eq. (F.6):

$$\begin{aligned}
&(1-\epsilon) A(\boldsymbol{\sigma}_{N-1}) \\
&\leq \frac{1}{2} \left[\sup_{\boldsymbol{f} \in \widetilde{\mathcal{F}}} \left[u(\boldsymbol{f}) + \sum_{b=1}^{B} f_b(z_N) \right] + \sup_{\boldsymbol{f} \in \widetilde{\mathcal{F}}} \left[u(\boldsymbol{f}) - \sum_{b=1}^{B} f_b(z_N) \right] \right] \\
&= \mathbb{E}_{\sigma_N} \left\{ \sup_{\boldsymbol{f} \in \widetilde{\mathcal{F}}} \left[u(\boldsymbol{f}) + \sigma_N \sum_{b=1}^{B} f_b(z_N) \right] \right\}
\end{aligned} \tag{F.7}$$

Since Eq. (F.7) holds for every $\epsilon > 0$, we have that:

$$A(\boldsymbol{\sigma}_{N-1}) \leq \mathbb{E}_{\sigma_N} \left\{ \sup_{\boldsymbol{f} \in \widetilde{\mathcal{F}}} \left[u(\boldsymbol{f}) + \sigma_N \mathbf{1}^T \boldsymbol{f}(z_N) \right] \right\} \tag{F.8}$$

Repeating this process for the remaining σs will eventually yield the result of this lemma.

□

APPENDIX G: PROOF OF THEOREM 4

Proof. Consider the function spaces:

$$\mathcal{G} \triangleq \{\boldsymbol{g} : (x,l) \mapsto [\mu_1(l)f_1(x), ..., \mu_B(l)f_B(x)]^T,\ \boldsymbol{\mu} \in \mathcal{M},\ \boldsymbol{f} : \mathcal{X} \mapsto \mathbb{R}^B\}$$

$$\Psi \circ \mathcal{G} \triangleq \left\{ \Psi(\boldsymbol{g}(\cdot)) : (x,l) \mapsto \frac{1}{B}\sum_{b=1}^{B} \Phi_\rho(g_b(x,l)),\ \boldsymbol{g} \in \mathcal{G} \right\}$$

Notice that, since $\Phi_\rho(u) \in [0,1]$ for $\forall u \in \mathbb{R}$, also $\Psi(\boldsymbol{g}(x,l)) \in [0,1]$ for $\forall \boldsymbol{g} \in \mathcal{G}$, $x \in \S$ and $l \in \mathbb{N}_G$. Hence, from Theorem 3.1 of [79], for fixed (independent of Q) $\rho > 0$ and for any $\delta > 0$ and any $\boldsymbol{g} \in \mathcal{G}$, with probability at least $1-\delta$, it holds that:

$$\mathbb{E}\left\{\Psi(\boldsymbol{g}(x,l))\right\} \leq \widehat{\mathbb{E}}_Q\left\{\Psi(\boldsymbol{g}(x,l))\right\} + 2\Re_N(\Psi \circ \mathcal{G}) + \sqrt{\frac{ln\frac{1}{\delta}}{2N}} \tag{G.1}$$

Where we define $\forall h : \mathcal{X} \times \mathbb{N}_C \mapsto \mathbb{R}$ and $\widehat{\mathbb{E}}_Q\left\{h(x,l)\right\} \triangleq \frac{1}{N}\sum_{n=1}^{N} h(x_n, l_n)$. Since $[u < 0] \leq \Phi_\rho(u)$ for $\forall u \in \mathbb{R},\ \rho > 0$, it holds that:

$$\frac{1}{B} d\left(\operatorname{sgn} \boldsymbol{f}(x), \boldsymbol{\mu}(l)\right) = \frac{1}{B}\sum_{b=1}^{B} \left[\mu_b(l) f_b(x) < 0\right] \leq \frac{1}{B}\sum_{b=1}^{B} \Phi_\rho\left(\mu_b(l) f_b(x)\right) = \Psi(\boldsymbol{g}(x,l))$$

Thus, we have the following:

$$er(\boldsymbol{f}, \boldsymbol{\mu}) \triangleq \mathbb{E}\left\{\frac{1}{B} d(\operatorname{sgn} \boldsymbol{f}(x), \boldsymbol{\mu}(l))\right\} \leq \mathbb{E}\left\{\Psi(\boldsymbol{g}(x, l))\right\} \tag{G.2}$$

Due to Eq. (G.2), with probability at least $1-\delta$, Eq. (G.1) becomes now:

$$er(\boldsymbol{f}, \boldsymbol{\mu}) \leq \widehat{\mathbb{E}}_Q\left\{\Psi(\boldsymbol{g}(x, l))\right\} + 2\Re_N(\Psi \circ \mathcal{G}) + \sqrt{\frac{ln\frac{1}{\delta}}{2N}} \tag{G.3}$$

Now due to the fact that $\Psi(\cdot)$ is $\frac{1}{B\rho}$ - Lipschitz continuous w.r.t $\|\cdot\|_1$ and Lemma 1, we have:

$$\widehat{\Re}_Q\left(\Psi \circ \mathcal{G}\right) \leq \frac{1}{B\rho} \widehat{\Re}_Q\left(\|\mathcal{G}\|_1\right)$$

Also, since $\boldsymbol{\mu} : \mathbb{N}_G \mapsto \mathbb{H}^B$, we have $\widehat{\Re}_Q\left(\|\mathcal{G}\|_1\right) = \widehat{\Re}_Q\left(\left\|\bar{\mathcal{F}}\right\|_1\right)$, where $\bar{\mathcal{F}}$ is defined in Eq. (4.18), we get:

$$\widehat{\Re}_Q\left(\Psi \circ \mathcal{G}\right) \leq \frac{1}{B\rho} \widehat{\Re}_Q\left(\left\|\bar{\mathcal{F}}\right\|_1\right)$$

Now due to Lemma 2, we have $\widehat{\Re}_Q\left(\left\|\bar{\mathcal{F}}\right\|_1\right) \leq \widehat{\Re}_Q\left(\mathbf{1}^T \bar{\mathcal{F}}\right)$, by taking expectations on both sides w.r.t Q, the above inequality becomes:

$$\Re_N\left(\Psi \circ \mathcal{G}\right) \leq \frac{1}{B\rho}\Re_N\left(\mathbf{1}^T\bar{\mathcal{F}}\right) \tag{G.4}$$

Substitute Eq. (G.4) into Eq. (G.3):

$$er(\boldsymbol{f}, \boldsymbol{\mu}) \leq \widehat{\mathbb{E}}_Q\{\Psi(\boldsymbol{g}(x,l))\} + \frac{2}{B\rho}\Re_N(\mathbf{1}^T\bar{\mathcal{F}}) + \sqrt{\frac{ln\frac{1}{\delta}}{2N}} \tag{G.5}$$

From the optimization problem in Eq. (4.6), we note that *Supervised Hash Learning (*SHL) is utilizing the hypothesis spaces defined in Eq. (4.18) and Eq. (4.19). Note the fact that each hash function of *SHL is determined by the data independent of the others.

By considering the Representer Theorem [97], we have $w = \sum_{n=1} \alpha_n \Phi_{\boldsymbol{\theta}}(x_n),\ \boldsymbol{\alpha} \in \mathbb{R}^N$. The aforementioned equation implies: $f(x) = \sum_n \alpha_n k_{\boldsymbol{\theta}}(x, x_n) + \beta$ and $\|w\|^2_{\mathcal{H}_{\boldsymbol{\theta}}} = \boldsymbol{\alpha}^T \boldsymbol{K}_{\boldsymbol{\theta}} \boldsymbol{\alpha}$. Here $\boldsymbol{K}_{\boldsymbol{\theta}}$ is the training data's kernel matrix.

Hence, $\mathcal{F}$ can be re-expressed as:

$$\mathcal{F} = \left\{ f : x \mapsto \sum_n \alpha_n k_{\boldsymbol{\theta}}(x, x_n) + \beta,\ \beta \in \mathbb{R},\ \boldsymbol{\alpha} \in \Omega_{\boldsymbol{\alpha}}(\boldsymbol{\theta}),\ \boldsymbol{\theta} \in \Omega_{\boldsymbol{\theta}} \right\}$$

where $\Omega_{\boldsymbol{\alpha}}(\boldsymbol{\theta}) \triangleq \{\boldsymbol{\alpha} \in \mathbb{R}^N :\ \boldsymbol{\alpha}^T \boldsymbol{K}_{\boldsymbol{\theta}} \boldsymbol{\alpha} \leq R^2\}$.

First of all, let's upper bound the Rademacher Complexity of *SHL's hypothesis space:

$$\widehat{\Re}_Q(\mathbf{1}^T\bar{\mathcal{F}}) = \frac{1}{N}\mathbb{E}_{\boldsymbol{\sigma}}\left\{\sup_{\boldsymbol{f}\in\bar{\mathcal{F}}}\sum_n \sigma_n \sum_{b=1}^{B} f_b(x_n)\right\} =$$
$$= \frac{1}{N}\mathbb{E}_{\boldsymbol{\sigma}}\left\{\sup_{f_b\in\mathcal{F}, b\in\mathbb{N}_B}\sum_b\sum_n \sigma_n f_b(x_n)\right\} =$$
$$= \sum_b \frac{1}{N}\mathbb{E}_{\boldsymbol{\sigma}}\left\{\sup_{f_b\in\mathcal{F}}\sum_n \sigma_n f_b(x_n)\right\} = B\widehat{\Re}_Q(\mathcal{F}) \tag{G.6}$$

Next, we will upper bound $\widehat{\Re}_Q(\mathcal{F})$:

$$\widehat{\Re}_Q(\mathcal{F}) = \frac{1}{N}\,\mathbb{E}_{\boldsymbol{\sigma}}\left\{\sup_{f\in\mathcal{F}}\sum_n \sigma_n f(x_n)\right\} =$$
$$= \mathbb{E}_{\boldsymbol{\sigma}}\left\{\sup_{\boldsymbol{\alpha}\in\Omega_{\boldsymbol{\alpha}}(\boldsymbol{\theta}),\boldsymbol{\theta}\in\Omega_{\boldsymbol{\theta}}} \boldsymbol{\alpha}^T\boldsymbol{K}_{\boldsymbol{\theta}}\boldsymbol{\alpha} + \sup_{\beta\in\mathbb{R}}\sum_n \sigma_n\beta\right\} \leq$$
$$\leq \frac{R}{N}\,\mathbb{E}_{\boldsymbol{\sigma}}\left\{\sup_{\boldsymbol{\theta}\in\Omega_{\boldsymbol{\theta}}}\sqrt{\boldsymbol{\alpha}^T\boldsymbol{K}_{\boldsymbol{\theta}}\boldsymbol{\alpha}}\right\} = \frac{R}{N}\,\mathbb{E}_{\boldsymbol{\sigma}}\left\{\sqrt{\sup_{\boldsymbol{\theta}\in\Omega_{\boldsymbol{\theta}}}\boldsymbol{\theta}^T\boldsymbol{u}}\right\} \tag{G.7}$$

where $\boldsymbol{u}\in\mathbb{R}^M$ such that $u_m \triangleq \boldsymbol{\sigma}^T K_m\boldsymbol{\alpha}$. The above inequality holds because of Cauchy-Schwarz inequality. Additionally, $\mathbb{E}_{\boldsymbol{\sigma}}\left\{\sup_{\beta\in\mathbb{R}}\sum_n \sigma_n\beta\right\} = 0$ since β is bounded.

By the definition of the dual norm, if $p' \triangleq \frac{p}{p-1}$, we have:

$$\sup_{\boldsymbol{\theta}\in\Omega_{\boldsymbol{\theta}}}\boldsymbol{\theta}^T\boldsymbol{u} = \|\boldsymbol{u}\|_{p'} \tag{G.8}$$

Thus, Eq. (G.7) becomes:

$$
\begin{aligned}
&\widehat{\Re}_Q(\mathcal{F}) \leq \frac{R}{N}\,\mathbb{E}_{\boldsymbol{\sigma}}\left\{\sqrt{\|\boldsymbol{u}\|_{p'}}\right\} = \\
&= \frac{R}{N}\,\mathbb{E}_{\boldsymbol{\sigma}}\left\{\left[\sum_{m=1}^{M}(\boldsymbol{\sigma}^T K_m \boldsymbol{\sigma})^{p'}\right]^{\frac{1}{2p'}}\right\} \leq \\
&\leq \frac{R}{N}\left[\sum_{m}\mathbb{E}_{\boldsymbol{\sigma}}\left\{(\boldsymbol{\sigma}^T K_m \boldsymbol{\sigma})^{p'}\right\}\right]^{\frac{1}{2p'}}
\end{aligned}
$$

The above inequality holds because of Jensen's Inequality. By the Lemma 5 from [59], the above expression is upper bounded by:

$$
\begin{aligned}
&\frac{R}{N}\left[\sum_{m}(p')^{\frac{p'}{2}}\left(\operatorname{trace}\{K_m\}\right)^{\frac{p'}{2}}\right]^{\frac{1}{2p'}} = \\
&= \frac{R}{N}(p')^{\frac{1}{4}}\left[\sum_{m}\left[\operatorname{trace}\{K_m\}\right]^{\frac{p'}{2}}\right]^{\frac{1}{2p'}}
\end{aligned}
\tag{G.9}
$$

Since $k_m(x, x') \leq r^2,\ \forall m \in \mathbb{N}_M,\ x \in \mathcal{X}$:

$$
\begin{aligned}
&\operatorname{trace}\{K_m\} \leq N r^2 \Rightarrow \left[\operatorname{trace}\{K_m\}\right]^{\frac{p'}{2}} \leq N^{\frac{p'}{2}} r^{p'} \Rightarrow \\
&\Rightarrow \left[\sum_{m}\left[\operatorname{trace}\{K_m\}\right]^{\frac{p'}{2}}\right]^{\frac{1}{2p'}} \leq M^{\frac{1}{p'}} N^{\frac{1}{4}} r^{\frac{1}{2}}
\end{aligned}
\tag{G.10}
$$

Thus, combine Eq. (G.9) and Eq. (G.10), we have:

$$\widehat{\Re}_Q(\mathcal{F}) \leq \frac{R}{N} q^{\frac{1}{4}} M^{\frac{1}{2p'}} N^{\frac{1}{4}} r^{\frac{1}{2}} = R \left(\frac{p'}{N^3} \right)^{\frac{1}{4}} \sqrt{r M^{\frac{1}{p'}}} \tag{G.11}$$

Combine Eq. (G.6) and Eq. (G.11):

$$\begin{aligned} &\widehat{\Re}_Q(\mathbf{1}^T \bar{\mathcal{F}}) \leq BR \left(\frac{p'}{N^3} \right)^{\frac{1}{4}} \sqrt{r M^{\frac{1}{p'}}} \Rightarrow \\ &\Rightarrow \Re_N(\mathbf{1}^T \bar{\mathcal{F}}) \leq BR \sqrt{r M^{\frac{1}{p'}} \sqrt{\frac{p'}{N^3}}} \end{aligned} \tag{G.12}$$

Finally, combine Eq. (G.5) and Eq. (G.12), one can generate the bound provided in Theorem 4.

□

LIST OF REFERENCES

[1] Philippe Aigrain, Hongjiang Zhang, and Dragutin Petkovic. Content-based representation and retrieval of visual media: A state-of-the-art review. *Multimedia Tools and Applications*, 3(3):179–202, 1996.

[2] F Gregory Ashby and Nancy A Perrin. Toward a unified theory of similarity and recognition. *Psychological Review*, 95(1):124, 1988.

[3] R. Balaji and R.B. Bapat. On euclidean distance matrices. *Linear Algebra and its Applications*, 424(1):108 – 117, 2007.

[4] Shumeet Baluja and Michele Covell. Learning to hash: Forgiving hash functions and applications. *Data Mining and Knowledge Discovery*, 17(3):402–430, 2008.

[5] G. Baudat and F. Anouar. Generalized discriminant analysis using a kernel approach. *Neural Computation*, 12(10):2385–2404, October 2000.

[6] Richard Ernest Bellman. *Dynamic Programming*. Dover Publications, Incorporated, 2003.

[7] Kristin P. Bennett. *Advances in Kernel Methods*. MIT Press, Cambridge, MA, USA, 1999.

[8] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, September 1975.

[9] Mikhail Bilenko, Sugato Basu, and Raymond J. Mooney. Integrating constraints and metric learning in semi-supervised clustering. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 81–88. ACM, 2004.

[10] Andrew Blake, Carsten Rother, Matthew Brown, Patrick Perez, and Philip Torr. Interactive image segmentation using an adaptive gmmrf model. In *Proceedings of the European Conference in Computer Vision (ECCV)*, May 2004.

[11] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the Annual Workshop on Computational Learning Theory*, pages 144–152, New York, NY, USA, 1992. ACM.

[12] Emmanuel J. Candès and Benjamin Recht. Exact matrix completion via convex optimization. *CoRR*, abs/0805.4471, 2008.

[13] Emmanuel J. Candès and Terence Tao. The power of convex relaxation: Near-optimal matrix completion. *CoRR*, abs/0903.1476, 2009.

[14] Qiong Cao, Yiming Ying, and Peng Li. Distance metric learning revisited. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD)*, pages 283–298, 2012.

[15] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[16] Lin Chen, Dong Xu, I.W.-H. Tsang, and Xuelong Li. Spectral embedded hashing for scalable image retrieval. *IEEE Transactions on Cybernetics*, 44(7):1180–1190, July 2014.

[17] Xi Chen, Weike Pan, James T. Kwok, and Jaime G. Carbonell. Accelerated gradient method for multi-task sparse learning problem. In *Proceedings of the International Conference on Data Mining (ICDM)*, pages 746–751. IEEE Computer Society, 2009.

[18] Yisong Chen, Guoping Wang, and Shihai Dong. Learning with progressive transductive support vector machine. In *Proceedings of the International Conference on Data Mining (ICDM)*, pages 67–74. IEEE Computer Society, 2002.

[19] Jian Cheng, Cong Leng, Peng Li, Meng Wang, and Hanqing Lu. Semi-supervised multi-graph hashing for scalable similarity search. *Computer Vision and Image Understanding*, 124:12 – 21, 2014.

[20] Sumit Chopra, Raia Hadsell, and Yann Lecun. Learning a similarity metric discriminatively, with application to face verification. In *Proceedings of the Computer Vision and Pattern Recognition (CVPR)*, pages 539–546. IEEE Press, 2005.

[21] T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, January 1967.

[22] A. Csillaghy, H. Hinterberger, and A. O. Benz. Content-based image retrieval in astronomy. *Information Retrieval*, 3(3):229–241, October 2000.

[23] Ritendra Datta, Dhiraj Joshi, Jia Li, and James Ze Wang. Image retrieval: Ideas, influences, and trends of the new age. *ACM Computing Surveys*, 40(2):5:1–5:60, May 2008.

[24] Jason V. Davis, Brian Kulis, Prateek Jain, Suvrit Sra, and Inderjit S. Dhillon. Information-theoretic metric learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 209–216. ACM, 2007.

[25] André C. P. L. F. de Carvalho and Alex A. Freitas. *A Tutorial on Multi-label Classification Techniques*, pages 177–195. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[26] Thomas G. Dietterich and Ghulum Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, 2(1):263–286, January 1995.

[27] Huyen Do, Alexandros Kalousis, Jun Wang, and Adam Woznica. A metric learning perspective of svm: On the relation of lmnn and svm. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 308–317, 2012.

[28] John Duchi and Yoram Singer. Efficient online and batch learning using forward backward splitting. *Journal of Machine Learning Research (JMLR)*, 10:2899–2934, December 2009.

[29] Mark Everingham, Luc Gool, Christopher K. Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338, June 2010.

[30] G. D. Finlayson. Color in perspective. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 18(10):1034–1038, Oct 1996.

[31] Alexander Gammerman, Volodya Vovk, and Vladimir Vapnik. Learning by transduction. *CoRR*, abs/1301.7375, 2013.

[32] Giorgio Giacinto. A nearest-neighbor approach to relevance feedback in content based image retrieval. In *Proceedings of the ACM International Conference on Image and Video Retrieval*, pages 456–463, 2007.

[33] Aristides Gionis, Piotr Indyk, and Rajeev Motwani. Similarity search in high dimensions via hashing. In *Proceedings of the International Conference on Very Large Data Bases (VLDB)*, pages 518–529, 1999.

[34] Amir Globerson and Sam T. Roweis. Metric learning by collapsing classes. In *Proceedings of Advanced Neural Information Processing Systems (NIPS)*, pages 451–458, 2005.

[35] Jacob Goldberger, Sam Roweis, Geoff Hinton, and Ruslan Salakhutdinov. Neighbourhood components analysis. In *Proceedings of the Neural Information Processing Systems Foundation (NIPS)*, pages 513–520. MIT Press, 2004.

[36] Robert L. Goldstone, Douglas L. Medin, and Jamin Halberstadt. Similarity in context. *Memory & Cognition*, 25(2):237–255, 1997.

[37] Yunchao Gong and Svetlana Lazebnik. Iterative quantization: A procrustean approach to learning binary codes. In *Proceedings of Computer Vision and Pattern Recognition (CVPR)*, pages 817–824, 2011.

[38] V. Gulshan, C. Rother, A. Criminisi, A. Blake, and A. Zisserman. Geodesic star convexity for interactive image segmentation. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010.

[39] Ulrike Hahn, Nick Chater, and Lucy B Richardson. Similarity as transformation. *Cognition*, 87(1):1 – 32, 2003.

[40] Trevor Hastie and Robert Tibshirani. Discriminant adaptive nearest neighbor classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 18(6):607–616, June 1996.

[41] Felix Hieber and Stefan Riezler. Bag-of-words forced decoding for cross-lingual information retrieval. In *Proceedings of Human Language Technologies and the North American Chapter of the ACL (HLT-NAACL)*, pages 1172–1182, 2015.

[42] Y. Hochberg and A. C. Tamhane. *Multiple Comparison Procedures*. John Wiley & Sons, Inc., New York, NY, USA, 1987.

[43] Jing Huang, S. Ravi Kumar, Mandar Mitra, Wei-Jing Zhu, and Ramin Zabih. Spatial color indexing and applications. *International Journal of Computer Vision*, 35(3):245–268, December 1999.

[44] Yinjie Huang, Michael Georgiopoulos, and Georgios C. Anagnostopoulos. Hash function learning via codewords. In *Proceedings of the European Conference on Machine Learning*

and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD), pages 659–674, 2015.

[45] Yinjie Huang, Cong Li, Michael Georgiopoulos, and Georgios C. Anagnostopoulos. Reduced-rank local distance metric learning. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD)*, 2013.

[46] David R. Hunter and Kenneth Lange. Quantile regression via an mm algorithm. *Journal of Computational and Graphical Statistics*, 9(1):60–77, Mar 2000.

[47] David R Hunter and Kenneth Lange. A tutorial on mm algorithms. *The American Statistician*, 58(1):30–37, 2004.

[48] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proceedings of ACM symposium on Theory of computing (STOC)*, pages 604–613, 1998.

[49] Masashi Inoue. On the need for annotation-based image retrieval. In *Proceedings of the workshop on information retrieval in context (IRiX)*, pages 44–46, 2004.

[50] Prateek Jain, Brian Kulis, Inderjit S. Dhillon, and Kristen Grauman. Online metric learning and fast similarity search. In *Proceedings of the Neural Information Processing Systems Foundation (NIPS)*, pages 761–768, 2009.

[51] T. Joachims. Transductive learning via spectral graph partitioning. In *Proceedings of the International Conference on Machine Learning (ICML)*. ACM, 2003.

[52] Thorsten Joachims. Transductive inference for text classification using support vector machines. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 200–209. ACM, 1999.

[53] Marius Kloft, Ulf Brefeld, Sören Sonnenburg, and Alexander Zien. lp-norm multiple kernel learning. *Journal of Machine Learning Research (JMLR)*, 12:953–997, July 2011.

[54] Matjaz Kukar, Igor Kononenko, and Si-Ljubljana. Reliable classifications with machine learning. In *Proceedings of the European Conference on Machine Learning (ECML)*, pages 219–231. Springer, 2002.

[55] B. Kulis, P. Jain, and K. Grauman. Fast similarity search for learned metrics. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 31(12):2143–2157, 2009.

[56] Brian Kulis and Trevor Darrell. Learning to hash with binary reconstructive embeddings. In *Proceedings of Advanced Neural Information Processing Systems (NIPS)*, pages 1042–1050, 2009.

[57] John Langfor, Lihong Li, and Tong Zhang. Sparse online learning via truncated gradient. *Journal of Machine Learning Research (JMLR)*, 10:777–801, 2009.

[58] Michael S. Lew, Nicu Sebe, Chabane Djeraba, and Ramesh Jain. Content-based multimedia information retrieval: State of the art and challenges. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 2(1):1–19, February 2006.

[59] Cong Li, Michael Georgiopoulos, and Georgios C. Anagnostopoulos. Multitask classification hypothesis space with improved generalization bounds. *Neural Networks and Learning Systems, IEEE Transactions on*, 26:1468–1479, 2015.

[60] Xi Li, Guosheng Lin, Chunhua Shen, Anton van den Hengel, and Anthony R. Dick. Learning hash functions using column generation. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 142–150, 2013.

[61] Hsin liang Chen and Edie M. Rasmussen. Intellectual access to images. *Library Trends*, 48(2), 1999.

[62] Guosheng Lin, Chunhua Shen, Qinfeng Shi, Anton van den Hengel, and David Suter. Fast supervised hashing with decision trees for high-dimensional data. In *Proceedings of Computer Vision and Pattern Recognition (CVPR)*, 2014.

[63] Guosheng Lin, Chunhua Shen, and Jianxin Wu. Optimizing ranking measures for compact binary code learning. In *Proceedings of European Conference on Computer Vision (ECCV)*, volume 8691, pages 613–627, 2014.

[64] Nikolas List and Hans Ulrich Simon. Svm-optimization and steepest-descent line search. In *Proceedings of the Conference on Computational Learning Theory (COLT)*, 2009.

[65] Wei Liu and Shih-Fu Chang. Robust multi-class transductive learning with graphs. In *Proceedings of the Computer Vision and Pattern Recognition (CVPR)*, pages 381–388. IEEE Press, 2009.

[66] Wei Liu, Junfeng He, and Shih-Fu Chang. Large graph construction for scalable semi-supervised learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 679–686, 2010.

[67] Wei Liu, Cun Mu, Sanjiv Kumar, and Shih fu Chang. Discrete graph hashing. In *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, pages 3419–3427, 2014.

[68] Wei Liu, Jun Wang, Rongrong Ji, Yu-Gang Jiang, and Shih-Fu Chang. Supervised hashing with kernels. In *Proceedings of Computer Vision and Pattern Recognition (CVPR)*, pages 2074–2081, 2012.

[69] Wei Liu, Jun Wang, Sanjiv Kumar, and Shih-Fu Chang. Hashing with graphs. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 1–8, 2011.

[70] X. Liu, Y. Mu, D. Zhang, B. Lang, and X. Li. Large-scale unsupervised hashing with shared structure learning. *IEEE Transactions on Cybernetics*, PP(99):1–1, 2014.

[71] S. Lloyd. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28(2):129–137, September 1982.

[72] Zhe-Ming Lu, Hans Burkhardt, and Sebastian Boehmer. Fast content-based image retrieval based on equal-average k-nearest-neighbor search schemes. In *Advances in Multimedia Information Processing*, volume 4261, pages 167–174. Springer Berlin Heidelberg, 2006.

[73] B. S. Manjunath and W. Y. Ma. Texture features for browsing and retrieval of image data. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 18(8):837–842, Aug 1996.

[74] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008.

[75] A.B. Markman and D. Gentner. Structural alignment during similarity comparisons. *Cognitive Psychology*, 25(4):431 – 467, 1993.

[76] Quinn McNemar. Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika*, 12(2):153–157, 1947.

[77] Douglas L Medin, Robert L Goldstone, and Dedre Gentner. Respects for similarity. *Psychological Review*, 100(2):254, 1993.

[78] S. Mika, G. Ratsch, J. Weston, B. Scholkopf, and K. R. Mullers. Fisher discriminant analysis with kernels. In *Proceedings of the IEEE Signal Processing Society Workshop on Neural Networks*, pages 41–48, Aug 1999.

[79] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of Machine Learning*. The MIT Press, 2012.

[80] Yadong Mu, Jialie Shen, and Shuicheng Yan. Weakly-supervised hashing in kernel space. In *Proceedings of Computer Vision and Pattern Recognition (CVPR)*, pages 3344–3351, 2010.

[81] Yu. Nesterov. Smooth minimization of non-smooth functions. *Mathematical Programming*, 103(1):127–152, 2005.

[82] Nam Nguyen and Yunsong Guo. Metric learning: A support vector approach. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD)*, pages 125–136, 2008.

[83] Yung-Kyun Noh, Byoung-Tak Zhang, and Daniel D. Lee. Generative local metric learning for nearest neighbor classification. In *Proceedings of the Neural Information Processing Systems Foundation (NIPS)*. MIT Press, 2010.

[84] Mohammad Norouzi and David J. Fleet. Minimal loss hashing for compact binary codes. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 353–360, 2011.

[85] Robert M Nosofsky. Attention, similarity, and the identification–categorization relationship. *Journal of Experimental Pychology: General*, 115(1):39, 1986.

[86] S. Omohundro. Efficient algorithms with neural networks behavior. *Complex Systems*, 1:273–347, 1987.

[87] Thomas H Painter, Jeff Dozier, Dar A Roberts, Robert E Davis, and Robert O Green. Retrieval of subpixel snow-covered area and grain size from imaging spectrometer data. *Remote Sensing of Environment*, 85(1):64–77, 2003.

[88] Neal Parikh and Stephen Boyd. Block splitting for distributed optimization. *Mathematical Programming Computation*, 6(1):77–102, 2014.

[89] Neal Parikh and Stephen Boyd. Proximal algorithms. *Foundations and Trends in Optimization*, 1(3), 2014.

[90] Kyoungup Park, Chunhua Shen, Zhihui Hao, and Junae Kim. Efficiently learning a distance metric for large margin nearest neighbor classification. In *Proceedings of AAAI Conference on Artificial Intelligence (AAAI)*, 2011.

[91] Maxim Raginsky and Svetlana Lazebnik. Locality-sensitive binary codes from shift-invariant kernels. In *Proceedings of Advanced Neural Information Processing Systems (NIPS)*, pages 1509–1517, 2009.

[92] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *Proceedings of Advanced Neural Information Processing Systems (NIPS)*, pages 1177–1184, 2008.

[93] A. Rakotomamonjy, R. Flamary, G. Gasso, and S. Canu. lp-lq penalty for sparse linear and sparse multiple kernel multi-task learning. *IEEE Transactions on Neural Networks*, 22:1307–1320, 2011.

[94] Volker Roth and Volker Steinhage. Nonlinear discriminant analysis using kernel functions. In *Proceedings of the Neural Information Processing Systems Foundation (NIPS)*, pages 568–574, 1999.

[95] Yong Rui, Thomas S. Huang, and Shih-Fu Chang. Image retrieval: Current techniques, promising directions, and open issues. *Journal of Visual Communication and Image Representation*, 10(1):39 – 62, 1999.

[96] Ruslan Salakhutdinov and Geoffrey Hinton. Semantic hashing. *International Journal of Approximate Reasoning*, 50(7):969–978, July 2009.

[97] Bernhard Schölkopf, Ralf Herbrich, and Alex J. Smola. A generalized representer theorem. In *Proceedings of the European Conference on Computational Learning Theory*, pages 416–426, 2001.

[98] Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10(5):1299–1319, July 1998.

[99] M. Schroder, H. Rehrauer, K. Seidel, and M. Datcu. Interactive learning and probabilistic retrieval in remote sensing image archives. *IEEE Transactions on Geoscience and Remote Sensing*, 38(5):2288–2298, Sep 2000.

[100] Emre Sefer and Carl Kingsford. Metric labeling and semi-metric embedding for protein annotation prediction. In *Research in Computational Molecular Biology*, pages 392–407. Springer, 2011.

[101] Gregory Shakhnarovich, Trevor Darrell, and Piotr Indyk. *Nearest-Neighbor Methods in Learning and Vision: Theory and Practice (Neural Information Processing)*. The MIT Press, 2006.

[102] Gregory Shakhnarovich, Paul Viola, and Trevor Darrell. Fast pose estimation with parameter-sensitive hashing. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 750–, 2003.

[103] Shai Shalev-Shwartz, Yoram Singer, and Andrew Y. Ng. Online and batch learning of pseudo-metrics. In *Proceedings of the International Conference on Machine Learning (ICML)*. ACM, 2004.

[104] Shai Shalev-Shwartz and Ambuj Tewari. Stochastic methods for l1-regularized loss minimization. *Journal of Machine Learning Research (JMLR)*, 12:1865–1892, 2011.

[105] Yuan Shi, Aurélien Bellet, and Fei Sha. Sparse compositional metric learning. In *Proceedings of AAAI Conference on Artificial Intelligence (AAAI)*, pages 2078–2084, 2014.

[106] Vikas Sindhwani and Haim Avron. High-performance kernel machines with implicit distributed optimization and randomization. *CoRR*, abs/1409.0940, 2014.

[107] Amit Singhal. Modern information retrieval: A brief overview. *IEEE Data Engineering Bulletin Issues*, 24(4):35–43, 2001.

[108] A. W. M. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain. Content-based image retrieval at the end of the early years. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 22(12):1349–1380, Dec 2000.

[109] Cees G.M. Snoek and Marcel Worring. Multimodal video indexing: A review of the state-of-the-art. *Multimedia Tools and Applications*, 25(1):5–35, 2005.

[110] Christoph Strecha, Alex Bronstein, Michael Bronstein, and Pascal Fua. Ldahash: Improved matching with smaller descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 34(1):66–78, January 2012.

[111] Michael J. Swain and Dana H. Ballard. Color indexing. *International Journal of Computer Vision*, 7(1):11–32, 1991.

[112] Partha Pratim Talukdar and Koby Crammer. New regularized algorithms for transductive learning. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD)*, pages 442–457. Springer, 2009.

[113] Daniel Tarlow, Kevin Swersky, Laurent Charlin, Ilya Sutskever, and Richard S. Zemel. Stochastic k-neighborhood selection for supervised and unsupervised learning. In *Pro-*

ceedings of the International Conference on Machine Learning (ICML), volume 28, pages 199–207, 2013.

[114] Yuan Tian, D. Lo, and Chengnian Sun. Information retrieval based nearest neighbor classification for fine-grained bug severity prediction. In *Working Conference on Reverse Engineering (WCRE)*, pages 215–224, Oct 2012.

[115] Adi Tom, O. K. Ewell, and Patricia Adi. High selectivity and accuracy with readware's automated system of knowledge organization. In *Proceedings of Text REtrieval Conference (TREC)*, 1999.

[116] A. Torralba, R. Fergus, and Y. Weiss. Small codes and large image databases for recognition. In *Proceedings of Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, 2008.

[117] Antonio Torralba, Rob Fergus, and William T. Freeman. 80 million tiny images: A large data set for nonparametric object and scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 30(11):1958–1970, November 2008.

[118] Lorenzo Torresani and Kuang C. Lee. Large margin component analysis. In *Proceedings of the Neural Information Processing Systems Foundation (NIPS)*, pages 1385–1392, 2007.

[119] Jeffrey K. Uhlmann. Satisfying general proximity / similarity queries with metric trees. *Information Processing Letters*, 40(4):175 – 179, 1991.

[120] Vladimir N. Vapnik. *Statistical Learning Theory*. Wiley, 1 edition, September 1998.

[121] J. Wang, S. Kumar, and S. F. Chang. Semi-supervised hashing for scalable image retrieval. In *Proceedings of the Computer Vision and Pattern Recognition (CVPR)*, pages 3424–3431, June 2010.

[122] Jun Wang, Alexandros Kalousis, and Adam Woznica. Parametric local metric learning for nearest neighbor classification. In *Proceedings of the Neural Information Processing Systems Foundation (NIPS)*, pages 1610–1618. MIT Press, 2012.

[123] Jun Wang, S. Kumar, and Shih-Fu Chang. Semi-supervised hashing for large-scale search. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 34(12):2393–2406, 2012.

[124] Jun Wang, Sanjiv Kumar, and Shih-Fu Chang. Sequential projection learning for hashing with compact codes. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 1127–1134, 2010.

[125] Qifan Wang, Luo Si, and Dan Zhang. Learning to hash with partial tags: Exploring correlation between tags and hashing bits for large scale image retrieval. In *Proceedings of European Conference on Computer Vision (ECCV)*, volume 8691, pages 378–392, 2014.

[126] Zhiyong Wang, Zheru Chi, and Dagan Feng. Fuzzy integral for leaf image retrieval. In *Proceedings of the IEEE International Conference on Fuzzy Systems*, volume 1, pages 372–377, 2002.

[127] Kilian Q. Weinberger, John Blitzer, and Lawrence K. Saul. Distance metric learning for large margin nearest neighbor classification. In *Proceedings of the Neural Information Processing Systems Foundation (NIPS)*. MIT Press, 2006.

[128] K.Q. Weinberger and L.K. Saul. Fast solvers and efficient implementations for distance metric learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 1160–1167. ACM, 2008.

[129] Yair Weiss, Antonio Torralba, and Robert Fergus. Spectral hashing. In *Proceedings of Advanced Neural Information Processing Systems (NIPS)*, pages 1753–1760, 2008.

[130] Rongkai Xia, Yan Pan, Hanjiang Lai, and Shuicheng Yan. Supervised hashing for image retrieval via image representation learning. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2014.

[131] Eric P. Xing, Andrew Y. Ng, Michael I. Jordan, and Stuart Russell. Distance metric learning with application to clustering with side-information. In *Neural Information Processing Systems Foundation (NIPS)*, pages 505–512. MIT Press, 2002.

[132] Liu Yang, Rong Jin, Rahul Sukthankar, and Yi Liu. An efficient algorithm for local distance metric learning. In *Proceedings of AAAI Conference on Artificial Intelligence (AAAI)*. AAAI Press, 2006.

[133] Peter N. Yianilos. Data structures and algorithms for nearest neighbor search in general metric spaces. In *Proceedings of the Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '93, pages 311–321, 1993.

[134] Yiming Ying, Kaizhu Huang, and Colin Campbell. Sparse metric learning via smooth optimization. In *Proceedings of the Neural Information Processing Systems Foundation (NIPS)*, pages 2214–2222, 2009.

[135] Yiming Ying and Peng Li. Distance metric learning with eigenvalue optimization. *Journal of Machine Learning Research (JMLR)*, 13(1):1–26, January 2012.

[136] Chun-Nam John Yu and Thorsten Joachims. Learning structural svms with latent variables. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 1169–1176, 2009.

[137] Yao-Liang Yu. Better approximation and faster algorithm using the proximal average. In *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, pages 458–466, 2013.

[138] A. L. Yuille and Anand Rangarajan. The concave-convex procedure. *Neural Computation*, 15(4):915–936, April 2003.

[139] Dell Zhang, Jun Wang, Deng Cai, and Jinsong Lu. Self-taught hashing for fast similarity search. In *Proceedings of the International Conference on Research and Development in Information Retrieval*, pages 18–25, 2010.

[140] Lei Zhang, Longbin Chen, Mingjing Li, and Hongjiang Zhang. Automated annotation of human faces in family albums. In *Proceedings of the ACM International Conference on Multimedia*, pages 355–358, 2003.

[141] Peichao Zhang, Wei Zhang, Wu-Jun Li, and Minyi Guo. Supervised hashing with latent factor models. In *Proceedings of the International Conference on Research and Development in Information Retrieval*, pages 173–182, 2014.

[142] Dengyong Zhou and Christopher J. C. Burges. Spectral clustering and transductive learning with multiple views. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 1159–1166. ACM, 2007.

[143] Pengfei Zhu, Qinghua Hu, Wangmeng Zuo, and Meng Yang. Multi-granularity distance metric learning via neighborhood granule margin maximization. *Information Sciences*, 282:321–331, October 2014.