UNIVERSITAT POLITÈCNICA DE CATALUNYA (UPC)

Master in Information Technology for Business Intelligence

Metric Learning Applied for Automatic large Scale Image Classification

THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF INFORMATION TECHNOLOGY FOR BUSINESS INTELLIGENCE

Student: Sahilu Wendeson Sahilu

Supervisor: Toon Calders (ULB)

Advisor: Salim Jouili (EURA NOVA)

UPC, Barcelona

September, 2014

Acknowledgement

First and foremost, praise and thanks to GOD, Almighty and Mother of SON Saint Virgin MARY, for showers of blessings throughout my life and to complete my thesis successfully.

I am heartily thankful to European Commission (ERASMUS MUNDUS) for giving me a chance to pursue my study in IT4BI program, and EURA NOVA for giving me the opportunity to do my thesis in their company.

Foremost, I would like to express my sincere gratitude to my supervisor Assistance Professor Toon Calders for continuous support of my thesis, for his patience, motivation, experience, immense knowledge, and enthusiasm. His guidance helped me in all the way of the research and writing of my thesis. My sincere thank also goes to my advisor Dr. Salim Jouili for offering me this area of research and leading me in all the way of this thesis.

Beside my advisor, I would like to thank Dr. Aymen Cherif, Dr. Jorge Luis Pérez Medina, Mr. Maher Mkadem, Mr. Hiroshi Leon, Mr. Sabri Skhiri and other EURA NOVA fellows for their encouragement, insightful comments, and hard questions. And I thank my IT4BI fellow, and childhood friends.

Last but not the least; I would like to thank my families and My Sintayehu for encouraging and supporting me spiritually.

Lastly, I offer my regards and blessings to all of those who supported me in any respect throughout my life, and during the completion of the project.

Sahilu Wendeson Sahilu

Barcelona, September 2014,

This thesis is dedicated to my Parents and My Sintayehu

For their endless love, support and encouragement

Table of Contents

Abstract

In the current Internet world, the numbers of digital images are growing exponentially. As a result, it is very tough to retrieve relevant objects for a given query point. For the past few years, researchers have been contributing different algorithms in the two most common machine learning categories to either cluster or classify images. There are several techniques of supervised classification images depending on the local or global feature representation of images, and on the metric used to calculate the distance (or similarity) between images. Recently many studies have shown the interest to learn a metric rather than use a simple metric a priori (e.g. Euclidean distance). This approach is described in the literature as *metric learning*. The main objective of this thesis is to use metric learning algorithm in the context of large-scale image classification.

In this project, we use a metric learning algorithm which is driven by the nearest neighbors approach and has a competence to improve the generic *k* Nearest Neighbor (kNN) machine learning algorithm. Even though we get significant improvement on the performance of classification, the computation is very expensive due to the large dimensionality of our input dataset. Thus, we use the dimension reduction technique to reduce dimension and computation time as well. Nevertheless, due to the size of the database, classifying and searching a given query point using metric learning algorithm alone exhaustively is intractable.

Therefore, to overcome this limitation *approximate nearest neighbor* (ANN) problem is proposed. In ANN, the data structure is allowed to get any data points which are closer to a query than a given radius, *r*. The best existing solution, *locality-sensitive hashing* (LSH), guarantees a high probability of collision for points which are close using a given metric space. There are LSH solutions using different similarity and metric distance measures. In this thesis, we use both Cosine similarity and *p-stable* distribution hashing solutions.

The major contribution of this thesis is to use a metric learning algorithm and formulate a fresh approach for hashing solutions to improve the classification performance.

Finally, we demonstrate our solutions' experimental results using different datasets to justify their performance improvement. All the experimental results show that, due to the effect of metric learning, our solutions improve classification accuracy compared with the traditional hashing techniques.

List of Tables

List of Figures

Chapter 1

1.1 Introduction

Human brains are extremely well-suited to extract a small number of relevant features from a collection of sensory data, and also classify them based on their similarity. However, it is infeasible to use human brain to classify images of large-scale database. In the current machine learning world, it is a fundamental challenge to develop and process internal representation of the image either to cluster or classify based on their similarity. There are several techniques of supervised classification of images depend on the representation of local features, and the metric distance to calculate the similarity (or distance) between images [1]. The first step has been getting a huge attention for the past more than a decade [2]. Recently, many studies have shown the interest to learn a metric rather than using a simple metric given a priori (e.g. Euclidean distance) [3]. This approach is described in the literature as *metric learning*. Even though there are few studies conducted using metric learning in the context of classification of large-scale image database, it is relatively new research topic. Due to this and other motivation factors, we are interested to jump into this area of research.

The main goal of metric learning is to learn the distance measure by using side information and also to optimize the parametric distance functions using Mahalanobis metric distance [4-8]. It considers the statistical distribution of the data points. It uses semi-definite programming to learn the distance; the details are covered in the next Chapter. There are a number of metric learning algorithms either to cluster or classify objects based on their similarity [9]. In this research, we focus on the algorithms which are driven by nearest neighbors approach to improve the generic *k* Nearest Neighbor (kNN) classification of

machine learning algorithm. There are some metric learning algorithms which are using nearest neighbor approach. We use both theoretical and experimental evaluations to choose one algorithm which is capable to accomplish the thesis objective, classification of large-scale image. The review of these algorithms is presented in the next Chapter.

Even though these metric learning algorithms improve the performance of traditional kNN classification, the computation is very expensive due to the large dimensionality of our input dataset [10]. Thus, we use dimension reduction technique in order to reduce the size of the dimension without much loss of information. In this thesis, we use the most well known and effective dimension reduction technique called *Principle Component Analysis* (PCA). The detail is presented in the next Chapter.

Even if the dimensionality of input data is reduced, but it is intractable to search relevant data points from a massive data collection for a given query point using metric learning algorithms in exhaustive way. Thus, metric learning algorithms alone does not perform well in large-scale database, as both storage overheads and distance computations become prohibitive [11, 12]. As a result, researchers proposed to use indexing techniques in order to enhance the performance of query nearest neighbor search. There are many efficient near(est) neighbor search algorithms specifically in the case of low dimensional data [13, 14]. However, in spite of decades of effort, these solutions suffer from a high query time and space complexity due to the growth of dimension *d*. This dimension phenomenon is often called "*the curse of dimensionality*" [15]. To overcome this limitation, researchers proposed a new research area called *approximate nearest neighbor* (ANN) *problem* [16]. In ANN, the data structure is allowed to get any data points which are closer to a query by a given radius, *r*. The best existing solution is *locality-sensitive hashing* (LSH), especially since the space and query time bounding is sub-quadratic, and the approximation factor is constant [16].

The basic idea of LSH is to formulate a hashing function which guarantees a high probability of collision for points which are close using a given metric space. Given such hash functions, it makes possible to retrieve nearest neighbors by hashing a query point. LSHs has been designed for different distance metric space; L_1 and L_2 distance, Cosine similarity, Hamming similarity, Jaccard index for set similarity and others [17]. For

instance, using Cosine similarity, hashing function(s) generates a sequence of compacted binary codes to represent each input by preserving the metric structure of input data. On the other hand, the representation is different in both L_1 and L_2 space hashing [18]. The details of these hashing techniques are presented in the next Chapter.

Generally, the aim of this research is to formulate a fresh hashing approach by breeding metric learning algorithm and metric space hashing technique to achieve large-scale image classification objective. Keep in mind this preamble; the following subsections will explain both the motivation factors and problem statements of the thesis.

1.2 Motivation

In the current data world the amount of data is increasing exponentially. Images are taking an important part of it. To exemplify this exponential data growth, we used Pingdom 2012 report [19]: Facebook adds 7 Petabytes of photos every month and 300 million new photos every day, 5 billion total number of photos are uploaded to Instagram since its start in October 2010 initial September 2012 and 58 photos are uploaded every second. Similarly, other web sites also have extensive contribution in the exponential growth of data in the current Internet world. Due to this data explosion, it makes very tough and computationally expensive to get appropriate search result for a given query. As a result of this, classification of large-scale datasets has been becoming the most dominant topic in the past decade: for instance; text classification[20], document classification [21], video classification [22], and image classification [23]. Image classification is one of the fundamental problems of computer vision and multimedia research and has been getting a huge attention for the past decade. This classification approach plays a vital role in the area of image search. To achieve this classification objective, kNN is a simple and easy supervised machine learning algorithm using distance or similarity measure is one approach. To answer this approach, a new subject called *metric learning* emerged. Although plenty of metric learning algorithms since 2002 have been contributed by different researchers, there is no single solution for this problem [8]. All the algorithms' performances are dependent on their area of application.

Even though these metric learning algorithms are existed, it is infeasible to directly use these algorithms to measure similarity and dissimilarity of data points in large-scale dataset. As a result, different structural solutions have been developed to index these dataset using a given distance metric space. LSH is a particularly valuable technique to solve the aforementioned problem, and reduces computational time drastically, at the cost of a small probability of failing to find the absolute closest match [24]. Therefore, we use this technique combine with metric learning algorithm to achieve the objective.

This area of research plays on enormous role in the current Information Technology world. In recent times, it has been applied to problems as diverse as link prediction in networks [25], state representation in reinforcement learning [26], music recommendation [27], learning hash functions [28], identity verification [29], webpage archiving [30], and partitioning [31], to name a few. There are also other fields' of applications like; Computer vision, Bioinformatics, and Information retrieval.

1.3 Problem Statement

Several techniques of supervised classification of images depend on the representation of local features of the image, and on the metric used to calculate the similarity (or distance) between the images. For the past few years, learning metric distance using Mahalanobis distance measure has been getting huge attention. As a result, lots of studies have been conducted in the context of classification. Each of these studies is area dependent. However, there is no a single solution due to different reasons. Therefore, this area of research is an open ended area.

Even though this metric learning approach has a competence to improve traditional kNN, it is intractable to use alone in a context of large-scale image database due to its computational complexity and other reasons. Moreover, it is very tough to get the nearest neighbor using metric learning algorithm from large-scale database [32]. Thus, both tree and hash based algorithms are candidates to address this bottleneck [33-35]. However, tree based indexing performs worst when the dimension is growing [18, 36, and 37]. Likewise, all the current space based indexing techniques fall not to answer the following demands [38]: high-dimensional data, structured input space and specialized similarity functions, and

availability of external supervision. On the other hand, hashing technique has a potential to answer the above demands [38]. Hashing is intelligent techniques to classify and keep similar data points in the same bucket. This helps also to get the nearest data points to a given query point within specific distance under a given metric space, specifically called *locality sensitive hashing* (LSH). Therefore, in this thesis, we use both metric learning algorithm and indexing technique to attain the objective.

1.4 Objectives and Contribution

Due to the reasons which are explained in the previous subsections, we are interested to dive into this area of research to set our fingerprint. The main objectives of this thesis are:-

I. To study and implement metric learning algorithm.
II. To study and implement dimension reduction technique
III. To study and implement LSH in different metric space
IV. To establish and implement machine learning evaluation techniques for both metric learning algorithm and hashing techniques.

The original contribution of the thesis are using metric learning algorithm and formulate a fresh approach for both Cosine and Euclidean space hashing to improve the classification performance. Moreover, we ensure that the selected metric learning algorithm has never used before with these hashing techniques.

1.5 Scope

To achieve all the above section listed objectives, there are ranges of topics and tasks have been taking place.

Generally, the scope of this research is listed as follow:-

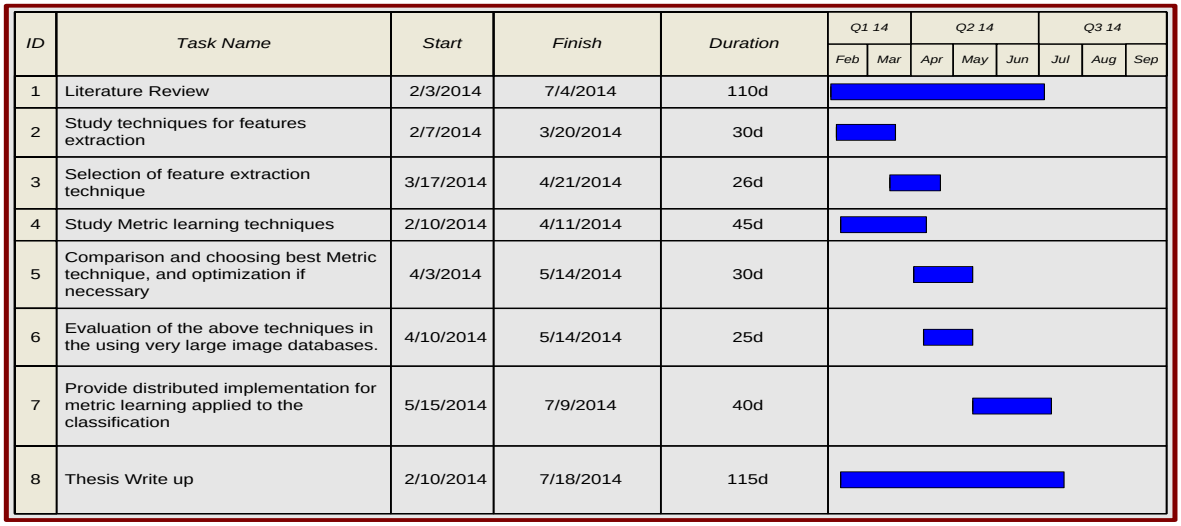
I. To study metric learning algorithms
II. To revise machine learning algorithms and evaluations techniques
III. To learn dimension reduction technique
IV. To study indexing techniques

Generally, the scope of this thesis is to answer all the objectives using the aforementioned areas of topics. Finally, evaluate the performance of the solutions and illustrate limitations and future directions.

1.6 Initial Planning

Although lots of change has been taking place over our initial plan, it is important to present in order to show the changes, limitations and other technical issues clearly. As shown in the Figure 1-1, the preliminary plan has eight tasks. The first task is "literature survey", to read the previous related works. It gets almost more than 5 months since it is very important task to understand the area clearly and propose solution. Meanwhile, other tasks also going on like: selection of feature extraction techniques, establishment of evaluation techniques to choose metric learning algorithm. The next part is to implement the selected algorithm and evaluate using different evaluation techniques. In addition, it is necessary to consider the nature of selected algorithm to provide a distributed implementation to apply in image classification. After some tests, the results should be better. If it won't be optimum, then it will be required to check the algorithm and its implementation. The thesis write up is also one of the major steps to document all the phases and make sure to understand the flow of this research. This task will take place after a couple of weeks till the last submission week.

Due to time and other reasons, this initial planning is revised after one and half months. The detail of the final plan is explained under final planning Chapter.



ID	Task Name	Start	Finish	Duration
1	Literature Review	2/3/2014	7/4/2014	110d
2	Study techniques for features extraction	2/7/2014	3/20/2014	30d
3	Selection of feature extraction technique	3/17/2014	4/21/2014	26d
4	Study Metric learning techniques	2/10/2014	4/11/2014	45d
5	Comparison and choosing best Metric technique, and optimization if necessary	4/3/2014	5/14/2014	30d
6	Evaluation of the above techniques in the using very large image databases.	4/10/2014	5/14/2014	25d
7	Provide distributed implementation for metric learning applied to the classification	5/15/2014	7/9/2014	40d
8	Thesis Write up	2/10/2014	7/18/2014	115d

Figure 1-1: Initial Plan

1.7 Thesis Structure

The rest of this thesis is organized into six chapters. Chapter 2 presents theoretical background of different mathematical formulations and reviews previous works. The third Chapter is dedicated to present what techniques we use to evaluate the algorithms, and how to apply these techniques; followed by result and discussion Chapter to present experimental results. Before conclusion and future direction Chapter, the final planning is presented to show the difference with the initial plan, and also the timetable of the thesis and factors which were the driving forces to change our initial plan. The last Chapter is conclusion and future direction, to recap the general thought of the research and present areas for future findings.

Chapter 2

2.1 Literature Review

This chapter of the thesis addresses the theoretical background and related work in the area of metric learning and hashing techniques. Furthermore, basic machine learning and mathematical backgrounds are explained, followed by related works of metric learning algorithms. Finally, hashing techniques under different metric space are presented.

2.2 Theoretical Background

In this image classification area of research, distance metric space is playing a great role in order to calculate the level of similarity and dissimilarity of data points, in a *pair-wise metric*. In the same way, the major contribution of metric learning is to improve this technique using different side information, and statistical distribution of data points. Before introducing and reviewing various metric learning algorithms, we will quickly review different techniques based on distance measure in machine learning. For instance, k-Nearest Neighbor classifier uses similarity computation to classify. There are different similarity measures metrics; Euclidean, Manhattan, Mahalanobis and others for numerical data points. Clearly, these metrics have different approach and also their application is varied. Euclidean distance is one of the simplest and most widely used metrics to measure dissimilarity (large distance) and similarity (small distance) between data points.

Fixed metrics, Euclidean distance, has constraint in considering the statistical distribution of the variables on the space and fails to capture idiosyncrasy of the data to improve the performance of classification, clustering and retrieval task [9]. Due to this reason, it produces unsatisfactory results. For example, in a given employee database table has: salary, age, service year and other attributes. In the context of Euclidean distance, it does

not consider any kind of correlation between attributes (service year Vs Salary) to calculate the distance between Employees. Due to this, it gives poor distance computation between employees. Due to this and other constraints, a new approach comes to exist called *metric learning*. The main goal of this approach is to improve the distance measure by incorporating side information to optimize the parametric distance function. Mahalanobis distance function is a perfect choice to handle the above requirements [39]. Thus, we are interested to use this approach in the context for similarity search in image databases.

When the image database is large-scale, computational complexity of searching similar images increases linearly. Specifically, in the current big data world, it is infeasible and very expensive to use distance measure exhaustively in the entire database to answer for a given query. Because of this, different data structure solutions have been existed to answer in a reasonable time. One of the solutions is using KD-tree and recursive hyperplane decomposition[1]. These techniques provide efficient result for low-dimensional data. However, the performance deteriorates when the dimension increases [40]. On the contrary, image has large dimension. Therefore, this solution does not fit to our requirement. The next and most dominant technique is hashing, *locality sensitive hashing* (LSH).

Locality sensitive hashing provides sub-linear search by hashing highly similar data points together in the same bucket. As a result, it gives a chance to get ANN data points towards the query data point.

Generally, the two main open research questions in the area of classification are: the performance of distance measure, and curse of dimensionality. Intuitively, it is required to consider the correlation of dimensions in which Mahalanobis distance measure has this feature. Furthermore, researchers have been contributing a number of metric learning algorithms for the past decade. The other research question is curse of dimensionality which makes the complexity very high, and tough to use common solutions like KD-tree. Therefore, this thesis is born under the context of the above two open questions.

The next subsection will present Mahanalobis distance, basic mathematical background, and previous related works of metric learning algorithms.

[1] www.cs.unc.edu/~lazebnik/fall09/large_scale_search.pptx
[2] http://web.stanford.edu/~boyd/cvxbook/

2.3 Mahalanobis Distance

Due to the reasons which are explained in the previous subsection, Mahalanobis distance measure has been used in the area of metric learning. Thus, this section presents its unique nature, mathematical formulation, and also challenges associated to metric learning.

The use of Mahalanobis metric removes several Euclidean metric limitations [41]:

a. It automatically accounts for the scaling of the coordinate axes

b. It corrects for correlation between the different features

c. It can provide curved as well as linear decision boundaries

The mathematical formulation of Mahalanobis distance measure is [39]:

$d_M\left(x, x^i\right) = \sqrt{(x - x^i)^T M (x - x^i)}$; Where M has to be positive semi-definite matrix, which is further explained in the next mathematical background section.

Due to these pleasant properties, it contributes a lot in the area of metric learning. However, it has two important challenges. These are [9]:

1. To maintain M as a positive semi-definite matrix during optimization process in an efficient way after projecting onto the positive semi-definite cone. It makes sure the optimization produces a positive semi-definite matrix

2. To learn a low-rank matrix instead of full-rank. Optimizing M subject to a rank constraint or regularization is *NP-hard* and thus cannot be carried out efficiently. If it carried out efficiently, it helps to decrease the computational time complexity. Like the above challenge, not all algorithms have the objective function to learn low-rank matrix.

2.4 Mathematical Background

In this section, we present definition and properties of *positive semi-definite* (PSD) matrices, followed by definitions of a metric space and how a well-defined metric over input data points can be obtained by transforming them into a different metric space.

2.4.1 Positive Definite Matrices

Semi-definite programming is playing a vital role in the area of metric learning. Before talking about SDPs, it is essential to present the basic linear algebra concept called positive semi-definite matrices[2].

Definition 1:- A symmetric matrix $\mathrm{M} \in \mathrm{S}^{\mathrm{n}\times\mathrm{n}}$ $(where\ M = M^T)$ is called positive semi-definite denoted by $\mathrm{M} \succcurlyeq 0$, if $\forall\, \vec{\mathrm{x}} \in \mathcal{R}^{\mathrm{n}}, \vec{\mathrm{x}}^{\mathrm{T}}\mathrm{M}\vec{\mathrm{x}} \geq 0$**.**

Definition 2:- if the condition in *Definition 1* is strictly inequality, then *M* is called *positive definite*$(M \succ 0)$.

The set of all positive semi-definite matrices denote by, $S_{+}^{n\times n} = \{M \in S^{n\times n} \mid M \succcurlyeq 0\}$. A single positive semi-definite matrix from *Definition 1*, $M \in S_{+}^{n\times n}$, has only non-negative eigenvalues. On the other hand, the set of positive definite matrices from *Definition 2* is denoted by $S_{++}^{n\times n} = \{M \in S^{n\times n} \mid M \succ 0\}$.

Lemma 1:-A Matrix *M* is positive semi-definite and positive definite, if and only if its eigenvalues are nonnegative $(\geq \mathbf{0})$ and positive (> 0) respectively.

From linear algebra, any symmetric matrix can be decomposed into a product of real matrices $M = V\Delta V^T$, where V contains the orthonormal eigenvectors of M and the diagonal matrix Δ contains eigenvalues which fulfilled *Lemma 1*. If the matrix is positive semi-definite, then it has upper and lower decomposition using *Cholesky decomposition*, and gives the following lemma.

Lemma 2:- A matrix $M \in s^{n\times n}$ is positive semi-definite matrix if and only if there exist a matrix $L \in \Re^{n\times n}$, such that $M = LL^T$ *(lower decomoposition)* and $G \in \Re^{n\times n}$, *such that* $M = G^T G$ *(Upper Decomposition)*

Using *Lemma 2*, we can rewrite Mahalanobis metric distance implicitly corresponds to computing Euclidean distance after transforming the data linearly by matrix *L*.

$$d_M(x, x') = \sqrt{(x - x')^T M (x - x')}$$
$$= \sqrt{(x - x')^T L^T L (x - x')} \quad = \sqrt{(Lx - Lx')^T (Lx - Lx')}$$

[2] http://web.stanford.edu/~boyd/cvxbook/

2.4.2 Metric Spaces

As discussed in the previous subsection, the main challenge in learning Mahalanobis learning distance is to maintain $M \in S_{+}^{n \times n}$ during the optimization process. This process of learning is called *semi-definite program (SPD)*. SPD is a convex optimization problem. Its main objective is to minimize the convex objective function to linear matrix and positive semi-definite matrix, Interested reader refer this [42]. The detail is not the objective of this thesis.

In the metric space distance there are properties of pseudo distance, for $M \in S_{+}^{n \times n}$ ensures that d_M satisfies the following properties [42] : $\forall x_i, x_j, x_k \in X$,

i. $d_M(x_i, x_j) \geq 0$ $(non - negativity)$,
ii. $d_M(x_i, x_j) = 0$ $(identity) \iff x_i = x_j$ $(Uniqueness)$,
iii. $d_M(x_i, x_j) = d_M(x_j, x_i)$ $(symmetry)$
iv. $d_M(x_i, x_j) + d_M(x_j, x_k) \geq d_M(x_i, x_k)$ $(triangular\ inequality)$,

Note; if the M $(M = I)$ is the identity matrix, Mahalanobis is equal to Euclidean distance. Otherwise decompose M using *Lemma 2* to project the data linearly using transformation matrix L.

2.5 Metric Learning

This subsection explains about the common ground of machine learning, and processing steps of metric learning algorithms including their key properties, and finally reviews previous related works.

2.5.1 Machine Learning Background

Machine Learning is a research field in computer science. Its main objective is to learn computers using machine learning algorithm(s) and input datasets. This input dataset is called training data. Based on the type of input datasets during the training, these algorithms are organized into different categories [43]. The most popular division is as follows:

- ***Supervised learning algorithms***: - are trained using training data which has a label for each data point. In other words, a data point is represented by a pair $(\vec{x}_i, y_i) \in X \times Y$, where $\vec{x}_i$ is input vector, and y_i corresponding label for the input vector. The assumption behind these pairs is that, there is some function $(f(\vec{x}_i) = y_i)$ which generates the label using these input vectors. This function f is unknown, thus it is a task of the machine learning algorithm to infer it from the training data. Finally, its quality is evaluated using the remaining datasets which is called test data. Generally speaking, supervised learning is a classification problem. Support vector machine [44, 45] and K-nearest neighbors [46] are examples of classification algorithms. These algorithms are used to answer problems like handwritten digitals recognition [47], document [21] and image classification, or face recognition [48]. Based on the reasons we have presented before, in this thesis, we use kNN classification.

- ***Unsupervised learning algorithms:*** - are trained using unlabelled data points in which the output is unknown for the input. One of the most common examples of unsupervised learning is clustering, used to group these points into meaningful categories. K-means [49] or spectral clustering [50] are examples of clustering algorithms. This thesis will not focus on clustering; we refer the interested reader to [51].

2.5.2 Metric Learning in a Nutshell

The notion of metric learning has started in earlier before two decades as per some earlier works [52-54]. However, the first time metric learning really emerged in 2002 with the pioneering work of [8] that formulates it as a convex optimization problem. It has been hot research areas for the past few years.

Additionally, in the past decade the advancement of convex optimization has been playing a great role and contributed a lot in the area of metric learning [42].

As discussed in the previous chapter, metric learning is a field to determine the optimum distance function, and achieve the objective either to classify or cluster data points. Due to the reasons which are mentioned in the previous section, Mahalanobis distance measure dominates the other techniques to measure pairwise distance using the information brought by training examples. There are different algorithms which are using different approaches mainly based on the way to learn *M* (PSD). As a result, their way of either classifying or clustering objects is different.

Nevertheless, all of the algorithms are sharing common processes which are shown in the Figure 2-1. A metric is learned from training data and plugged into an algorithm that outputs a predictor which can be classifier, clustering, or else. Finally, the algorithm will predict better than a predictor which is induced by a standard or non-learned metric [23]. Mainly, the resulting algorithm is used to improve metric-based algorithms like k-Nearest Neighbors (kNN) and also clustering algorithms such as K-Means, a ranking algorithm, etc [23].

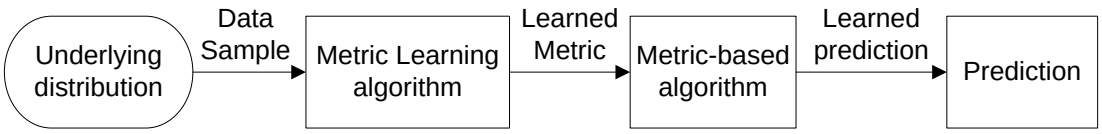


Figure 2-1: Common Process of Metric Learning [23]

Most of the current metric learning algorithms are very much competitive in the context of their performance to achieve specific problems. These algorithms have different applications and use different approaches as well. But in this thesis, we emphasize nearest neighbor driven metric learning algorithms which are reviewed in the next subsection.

2.5.3 Related Metric Learning Algorithms

Even though this area is new, lots of works have been going on under different application. This subsection presents a compressive review of selected supervised Mahalanobis distance learning methods of the literature, specific to k-nearest neighbor approach. kNN is one of the most widely used supervised learning. Thus, every data point belongs either of the given classes. These classes help to train the data. Under this context, all the data points which have identical class are called similar (*S*), and if not they are called dissimilar (*D*).

The first Mahalanobis distance learning method called *Mahalanobis Metric Clustering* (MMC) algorithm, aims at maximizing the sum distance between dissimilar (D) points while minimizing similar (S) data points in $\mathbb{R}^{m\times n}$ [8]. Its objective function is based on convex formulation with no regularization. For instance, pairs of points (x_i, x_j) in S have small squared distance between them; $Minimize_M \sum_{(x_i,x_j)\in S} \| x_i - x_j \|_M^2$. This is trivially solved with $M = 0$, which is not useful, and constraint $\sum_{(x_i,x_j)\in D} \| x_i - x_j \|_M \geq 1$ to ensure that M does not collapse the dataset into a single point. Here, if the information is explicit

to categorize pairs of points into D, no need to classify under *S;* otherwise, it may take all pairs it not to be in S. This gives the optimization problem:

$$\begin{aligned} &\min_{A} \ \sum_{(x_i,x_j)\in S} \| x_i - x_j \|_M^2 \\ &s.t \ \sum_{(x_i,x_j)\in D} \| x_i - x_j \|_M \geq 1, \\ &M \succcurlyeq 0, \end{aligned}$$

It used to improve the clustering performance, even if it gives local-optimal. This algorithm is used a simple projected gradient approach requiring the full eigenvalue decomposition of M (has to be PSD) at each iteration. This makes it intractable for medium and high-dimensional problems [9].

One of the groundwork of metric learning algorithms in the area of kNN approach is the idea of Neighborhood Component Analysis (NCA) introduced by Goldberger et al [5]. This algorithm proposes to use Mahalanobis distance measure to resolve two of serious drawbacks of kNN which are computational and modeling issues by learning a quadratic distance metric which optimizes the expected leave-one-out (LOO) classification error on the training data using stochastic neighbor selection rule [5]. Its main idea is to optimize the expected LOO error of a stochastic nearest neighbor classifier in the projection space induced by d_M [9]. This classifier in NCA used majority vote of nearby training examples. Particularly, for a given query, softmax probability distribution is used to retrieve examples from training set that favors nearby data points. They use the decomposition $M = L^T L$ and make linear transformation ($\vec{x} \to L\,\vec{x}$). This transformation used to minimize the expected classification error. Therefore, this algorithm considers only define the probability that x_i is the nearest neighbor of x_j by:

$$p_{ij} = \frac{e^{(-\|x_i - Lx_j\|_2^2}}{\sum_{l\neq i} e^{(-\|Lx_i - Lx_l\|_2^2}}, \quad p_{ii} = 0$$

The probability that x_i is correctly classified is: $p_i = \sum_{j:y_j=y_i} p_{ij}$

Learn distance by solving: $\max_{L} \sum_i p_i$

Unlike other methods, it is non-parametric, making no assumptions about the shape of the class distributions or the boundaries between them. The main limitation of this technique is non-convex and thus subject to local maxima.

To tackle the previous NCA drawback, a new algorithm known as Maximally Collapsing Metric Learning algorithm (MCML) is designed [55]. Its main objective is to find convex formulation of NCA. The main theoretical foundation is that if all points in the same class could be mapped into a single location, and all points other classes mapped to other locations. This algorithm approximates this scenario via a stochastic selection rule like NCA. However, like MMC, MCML requires costly projection onto the PSD cone.

Large Margin Nearest Neighbors (LMNN) is introduced by Weinberger et al [56], one of the most widely-used Mahalanobis distance metric for kNN classification using semi-definite programming, and also subject for many metric learning algorithm extensions. Likewise other related works, this algorithm improves kNN classification by learning the distance using Mahalanobis metric distance. The ground idea for this algorithm is NCA's LOO technique. However, the approach is different. NCA uses probability distribution. LMNN uses semi-definite programming. As shown in the figure below, LMNN theoretical foundation is to minimize the distance between data points which are in the same class, and maximize the distance by large margin between different classes (the "*impostors*"). Thus the name of the approach is *large margin nearest neighbor* (LMNN). It defines the constraints locally. Formally, the constraints are defined in the following way [9]:

$$S = \{(x_i, x_j): y_i = y_j \ and \ x_j \ belongs \ to \ the \ k - neighborhood \ of \ x_i\},$$
$$R = \{(x_i, x_j, x_k): (x_i, x_j) \in S, y_i \neq y_k\}.$$

The distance is learned using the following convex program:

$$\min_{M \in \mathbb{S}_+^d} \ (1-\mu) \sum_{(x_i,x_j)\in S} d_M^2(x_i, x_j) \ + \ \mu \sum_{i,j,k} \xi_{ijk}$$; Where ξ_{ijk} : is slack variable to mimic the effect of hinge loss in Support Vector Machine (SVM)

$$s.t. \ \ d_M^2(x_i, x_k) - \ d_M^2(x_i, x_j) \ \geq 1 - \xi_{ijk} \quad \forall (x_i, x_j, x_k) \in \mathcal{R}, \xi_{ijk}$$

Generally, this algorithm performs very well in practice, even if it sometimes prone to overfitting due to the absence of regularization, especially in high dimension [23]. Hence, we chose this algorithm to attain the objective of this research. One of the potential

weaknesses of LMNN is specifying target neighbors prior. To resolve this drawback, we use Euclidean distance. This approach is another branch of LMNN algorithm which is known as *Multi-pass LMNN*. For detail mathematical formulation of the algorithm, we recommend to refer the Appendix section of original paper [10].

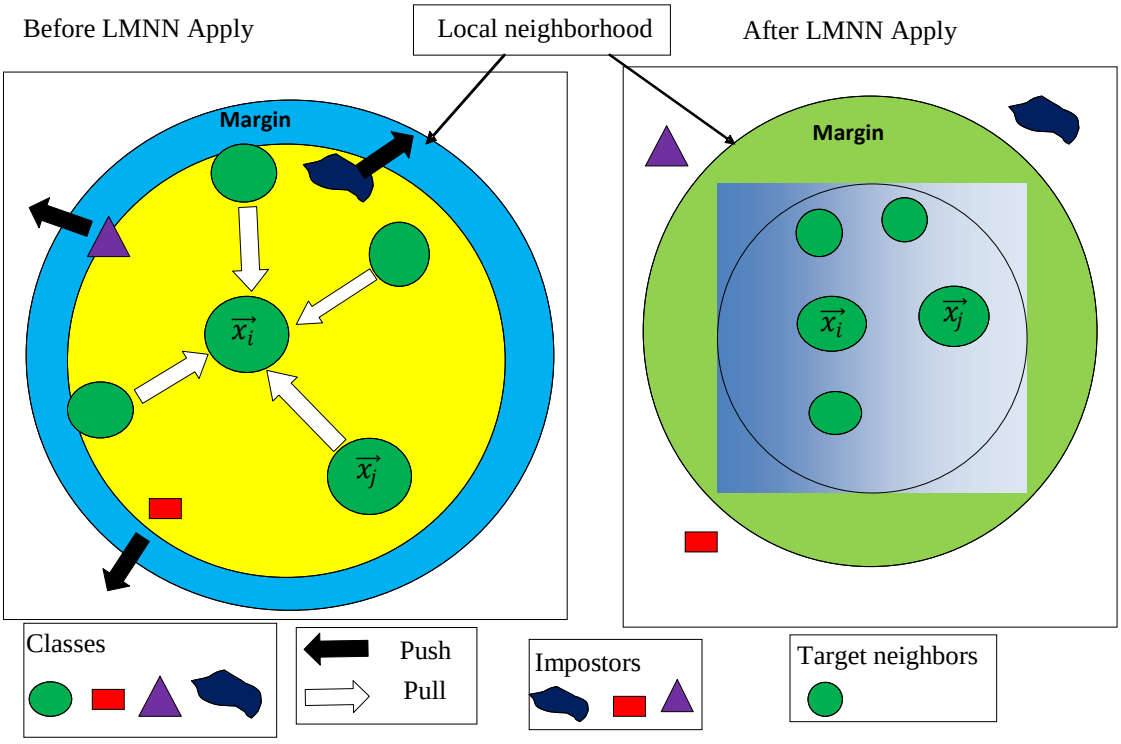


Figure 2-2: A schematic illustration of LMNN

There are five extensions designed to improve LMNN classification techniques. However, their computational complexity is expensive. These are multi-task LMNN [57], multiple metrics LMNN [58], Gradient-boosted LMNN [59], X^2-LMNN [60], and kernelized version of LMNN [61].

Even though the above metric learning algorithms are emerged to improve kNN classification, due to exponential growth of data, using metric learning algorithms alone is computationally very expensive and its time complexity increases linearly with the data size. To overcome these limitations, indexing (hashing) techniques are used to tackle this problem using data structure solution. In addition, dimension reduction technique is also used to reduce dimensions of input data. It helps to reduce computational time.

2.6 Dimension Reduction

Dimension reduction is the method used to map data into a lower dimensional space such that uninformative variance in the data is discarded, or such that a subspace in which the data lives is detected [62]. As presented in the previous chapter, curse of dimensionality is the main bottleneck to process massive data, especially in image classification. To tackle this problem, dimension reduction plays a great role to reduce the dimension of images in order to process in a reasonable time.

There are two major division of dimensional reduction; linear and nonlinear techniques [62]. *Principal component analysis* (PCA) is the most popular linear technique. Its main advantage is to reduce the dimension without much loss of information, and powerful technique to analyze the data and avoids overfitting [63]. Due to this nature, we use this technique in order to reduce the dimension of our input data which are images.

To illustrate how PCA is working with example, we use the following three dimension data: Age, Weight, and Height of human parameters. The distribution is shown in the following figure. The goal is to reduce the dimension without much loss of information.

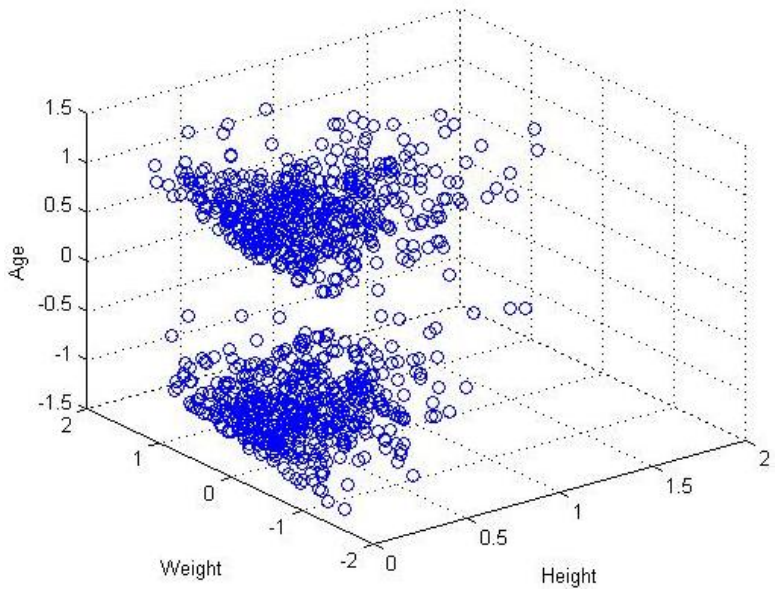


Figure 2-3:3-Dimensional data distribution

To reduce the dimension of the above data, we plug PCA algorithm. As a result, we get the result which is depicted in the following figure. The number of reduced dimension is two.

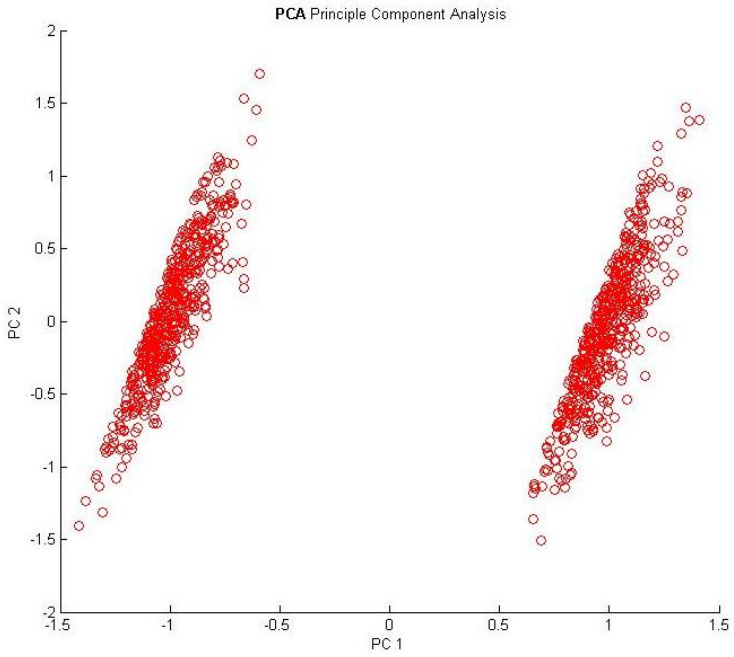


Figure 2-4: Reduced dimension using PCA

In General, we use PCA to reduce the dimension of the data before plugging into our algorithm. Moreover, preprocessing the input with PCA helps to reduce computational time plus avoid overfitting.

2.7 Similarity Search via hashing

In the current big data world, computationally linear search using any distance measure is very expensive and intractable. Thus, the nearest-neighbor query problem gets huge attention in a large variety of database applications using the context of similarity searching [38]. This problem is solved by using a technique called indexing to perform similarity search over high dimensional data, like image database, document collections, time-series databases, and others [36]. However, all the known techniques are dimensional dependent. For instance, KD tree data structures poorly when dimension is increases. If the dimension is getting large, its computational complexity grows to $O(N)$ [38]. Hence, this indexing technique is not better than linear search in large dimension, especially images.

On the other hand, building hash table which ensures high probability of collision for all closer points is another solution. This hashing technique minimizes the complexity and offers sub-linear time search. Moreover, it makes easy to find exact and also approximate

matches efficiently. Before presenting LSH in-depth, the following definitions are basics [36]:

Definition 3 (Nearest Neighbor Search) Given a set *P* of *n* object represented as points in a normed space M_p^d (Mahalanobis space) preprocess *P*, to answers queries by finding the point in *P* closest to a query point *q*

This *Definition 3* generalizes the natural case to return the nearest data points where $k > 1$ points which are closest to the query points. NNs approximate version is defined as follow using approximation factor.

Definition 4 (ε-Radius Based Nearest Neighbor Search) Given a set *P* of points in a space M_p^d, preprocess *P* to return a point $p \in P$ for any given query point *q*, such that: $d(q, p) \leq (1+\varepsilon)d(q,P')$, where $d(q,P')$, is the distance of *q* to the its closest point in *P*.

Definition 4 generalizes the nature of approximate kNNs problem to find $k > 1$ approximate nearest neighbors. It helps to find *k* points $p_1, \dots p_k$ such that the distance of p_k to the query *q* is at most (1+ ε) times the distance from closet point P' to q. In addition, it minimize the query time by sub-linear retrieval time even for high dimensional input data. Its query-time cost depends only linearly on the number of bits used [38].

2.7.1 Locality Sensitive Hashing

By using the benefits of approximate nearest neighbor a new indexing method is introduced which has sub-linear dependence on the data size even for high-dimensional data [36]. This technique is called *locality sensitive hashing* (LSH). It is randomized hashing framework. The key idea is to hash the data points using different hash functions to ensure collisions of close data points than those which are far apart. LSH is very simple, easy to implement and intelligent technique. Due to its remarkable characteristics, we use it in this project to accomplish the research objective. Intuitively, if all highly similar data points collide in the hash table (i.e., are assigned the same hash key), then at the query time, it is easy to get all closer data points. As shown in Figure 2-5, the hash keys consist of low-dimensional binary strings; and each data points is mapped to *b* bits. These bits are generated by *b* number of independent valid hash function(s).

Locality sensitive hashing was introduced by Indyk and Motwani [16] to devise main memory nearest neighbor search algorithm. It achieves $O\left(dn^{1/\varepsilon}\right)$- time for approximate nearest neighbor query over an *n*-point database, *d*-dimension, and ε- approximation factor.

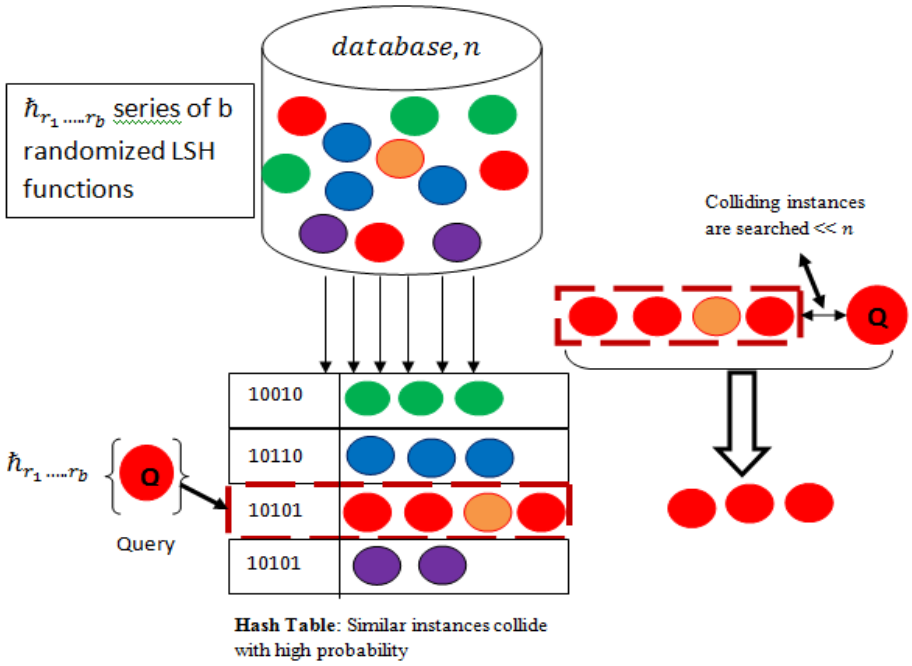


Figure 2-5: A schematic illustration of Locality Sensitive Hashing

As shown in the above figure, once all database items have been hashed into the hash table(s) by computing their unique signature (sequence of bits), the same randomized functions are applied to novel queries to give its signature. Using query's signature, search a given bucket which collides with this signature. As a result, it improves the exhaustive search from the entire database.

Suppose we have a database consisting of data points $x_1, x_2, \ldots, x_n$. The hash keys are constructed in order to give binary signature for each data points. These functions are equal to the number of bit, *b*. The functions are: $H = \{h_1, h_1, \ldots h_b\}$. Given an input query q, there are defined techniques to measure similarity to get most similar data points from the database.

Definition 5 (Nearest Neighbor-based LSH) the formulation of LSH functions is equating the collision probabilities with their similarity score; each hash function h_H from the distribution *H* must satisfy [38]:

$\Pr[h_H(x_i) = h_H(x_j)] = sim(x_i, x_j)$, Where $sim(x_i, x_j) \in [0,1]$ is the similarity function of interest.

Using *Definition 5*, the goal is to retrieve database item within a given radius of the query.

Generally, it is intuitive that LSH's query time complexity depends only linearly on the number of bits used [38].

2.7.2 Metric Space Hashing

Based on the main idea of LSH; for different metric distance and similarity measures, hashing functions has been designed [64]. All these hash functions ensure the generic characteristics of LSH. In this thesis, we plan to use both the Cosine similarity, and *p-stable* (Euclidean metric distance) based hashing. Moreover, we use metric learning algorithm to learn the aforementioned implementation of hashing. In the following subsection, these techniques are presented.

2.7.2.1 Cosine Similarity Based Hash Functions

The traditional unsupervised LSH function would generate random hyperplane to separate instances randomly [38]. The number of hashing function depends on the number of bits to encode the data point. The generic hash function is dot product of random generated hyperplane with data point. The function is as follows [32]:

$$h_r(x) = \begin{cases} 1, & if\ r^T x \geq 0 \\ 0, & otherwise \end{cases}$$

Where, vector *r* is a hyperplane chosen at random Gaussian distribution with zero mean and unit variance, zero-mean multivariate Gaussian $\mathcal{N}(0,1)$ of the same dimension as the input *x* as shown in the Figure 2-6. This hash function satisfies *Definition 5*.

For the following two vectors $x_i, and\ x_j$, using Goemans and Williamson [65] in their rounding scheme for the semi-definite programming relaxation of MAX-CUT [64],

$$\Pr\left[\mathrm{h_r}(\mathrm{x_i}) = \mathrm{h_r}(\mathrm{x_j})\right] = 1 - \frac{1}{\pi} cos^{-1}\left(\frac{x_j^T x_j}{\sqrt{|x_i||x_j|}}\right),$$

$$the\ Cosine\ is\ the\ degree\ between\ the\ two\ vectors\ (x_i, x_j)$$

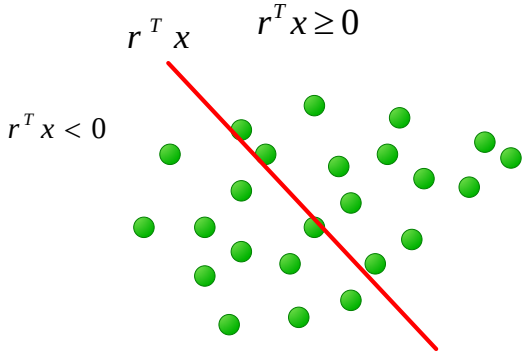


Figure 2-6:LSH for dot Product

Randomized hash functions using learned distance ensure similar class objects become more likely to collide, while dissimilar class objects become less likely to collide. The hashing is learned by using the output of LMNN, matrix M for a metric learned, using *Lemma 2*, $M = G^T G$. Thus, generate the following learned hash functions, $h_{r,M}$ [32]:

$$h_{rM}(x) = \begin{cases} 1, & if\ r^T Gx\ \geq 0 \\ 0, & otherwise \end{cases}$$

For this learned LSH hash function, *Definition 5* by parameterizing the hash function by *r* and *M*, the following relationship obtains [32]. Similar to random hyperplane based hash function scheme of probability, using Goemans and Williamson [65],

$$\Pr\left[\mathrm{h_{r,M}}(\mathrm{x_i}) = \mathrm{h_{r,M}}(\mathrm{x_j})\right] = 1 - \frac{1}{\pi} cos^{-1}\left(\frac{x_j^T M x_j}{\sqrt{|Gx_i||Gx_j|}}\right),$$

As presented in the aforementioned paragraphs, each bucket of the LSH is represented by a sequence of bits ($\{0,1\}$). There are different methods to measure the distance of these binary representations. One of the most well-known and cheap method is *Hamming distance*,d_H under *d*-dimensional hamming space [32]. If we have two points which are represented in a sequence of binary bits ($x, y \in H^d$), the Hamming Distance[3], $d_H(x, y)$ between these data points is the number of positions at which the corresponding

[3] http://i.cs.hku.hk/~hubert/teaching/c8601_2009/notes6.pdf

strings differ; $d_H(x,y) := |\{i{:}x_i \neq y_i\}|$. This distance measure computational complexity is depends on the number of bits we used, $O(b)$.

Based on *Definition 5*, the nearest neighbors search using the hamming distance; for instance if we have a set of P of n points in Hamming Space H^d, and a query $q \in H^d$, a nearest neighbors this query in P is all the $p \in P$ which has minimum Hamming distance $d_H(p,q)$ under a given radius, r. The following figure illustrates the structure of hash table and how the nearest neighbor search is retrieved for a given query point.

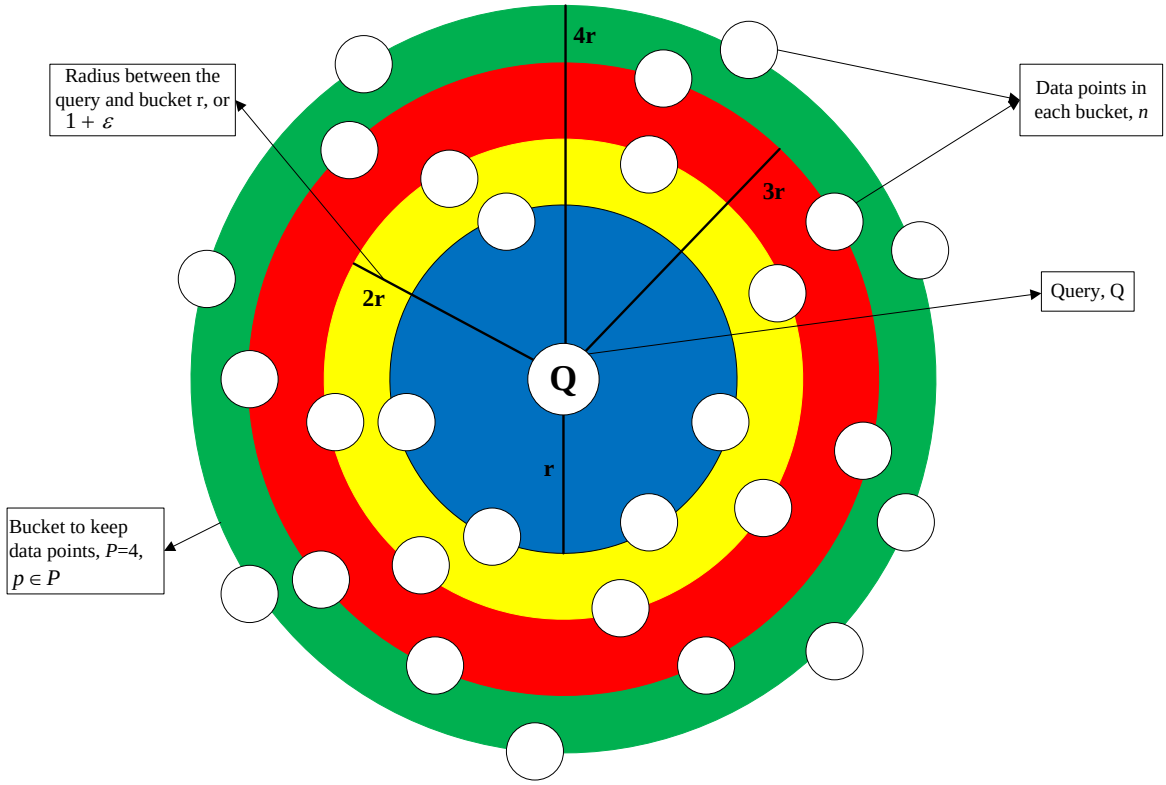


Figure 2-7: Schematic illustration of nearest neighbor search in LSH

As shown in the above figure, each circle ("Big") is represented bucket to keep the data points ("Small" Circle). The query point (Center) has variable distance with each of the buckets. To access the nearest neighbors for the query point out of this hash table, the following techniques are used[2]:

- ✓ *Naive Method* :- a query point q, the Hamming Distance $d_H(p,q)$ for each $p \in P$ is computed in time $O(d)$. Hence, it takes $O(nd)$ time to find a nearest neighbor. The computational time complexity is linear with the number n of points in P. It is very expensive.

- ✓ *Approximate Nearest Neighbor Search*:- approximation ratio $\boldsymbol{\varepsilon} > 1$, a $(1+\boldsymbol{\varepsilon})$-nearest neighbor for q in P is a point $\boldsymbol{p} \in \boldsymbol{P}$ such that the Hamming distance $d_H(q, p) \leq (1+\boldsymbol{\varepsilon})d_H(q,\boldsymbol{P})$, where $d_H(q, p)$ is a nearest neighbor of q. It makes possible to return an ANNs for a given query in sub-linear time $\boldsymbol{O}(\boldsymbol{n})$.

- ✓ *Approximate Range Search*:- A range parameter or radius $r > 0$ and $\boldsymbol{\varepsilon} > 1$ are given. The search with specified range r and approximation ratio $\boldsymbol{\varepsilon}$ does the following: if there is some point $\boldsymbol{p}^* \in \boldsymbol{P}$ such that $d_H(q, p^*) \leq r$, return a point $\boldsymbol{p} \in \boldsymbol{P}$ that satisfies $d_H(q, p) \leq \boldsymbol{\varepsilon} r$. If there is no point $\boldsymbol{p}^* \in \boldsymbol{P}$ such that $d_H(q, p^*) \leq r$, approximate range search could still return a point $\boldsymbol{p} \in \boldsymbol{P}$ that satisfies:

$$r < d_H(q, p) \leq \boldsymbol{\varepsilon} r.$$

In this thesis, the company side wants to implement and test the performance of *approximate nearest neighbor search* of our hashing solution. Therefore, we use ANN search in all experimental test which is presented under result and discussion Chapter.

Finally, to demonstrate the effect of metric learning algorithm on the hash function, the following image database is used (Figure 2-8[4]).

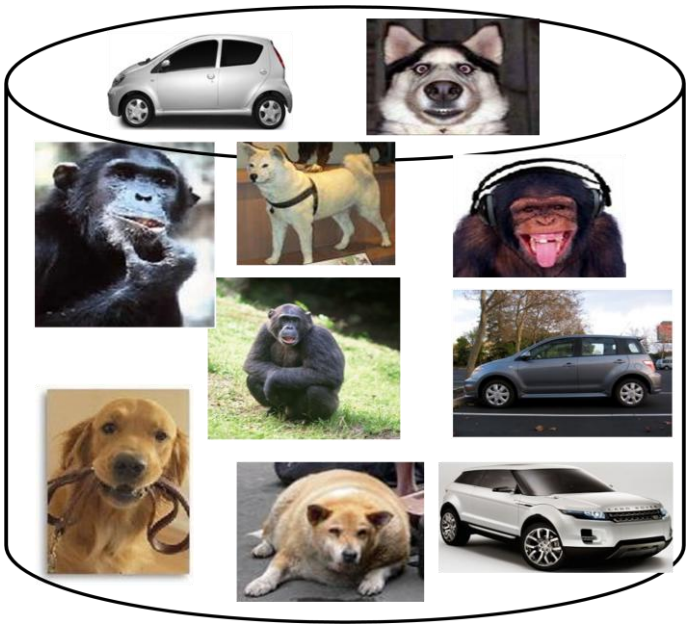
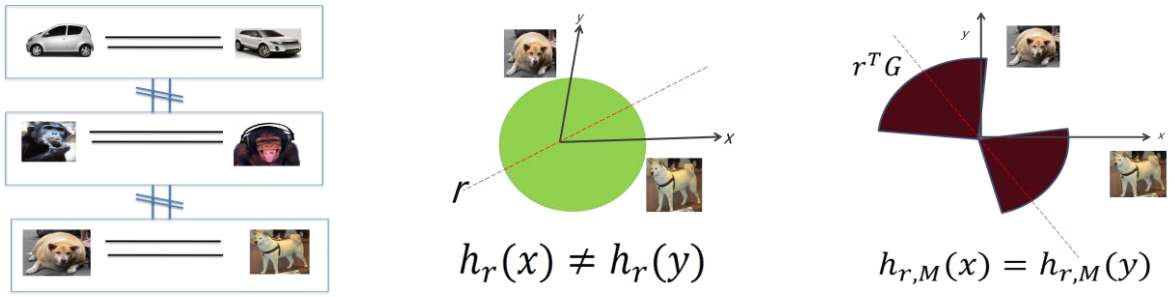


Figure 2-8: Image database

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



(a) Paired constraints (b) Hash function, original (c) Supervised hash function

Figure 2-9:Effect of metric learning, and unlearned Versus learned hash function

When LMNN is applied on the database, paired constraints is obtained, images under similar class pull each other (straight line) while dissimilar class are push each other (crossed out line). Therefore, as shown in the Figure 2-9 (a) images under similar class are much closer than dissimilar class.

If randomized LSH functions ($r^T x$) is used on the original database, it puts images together under the original distance. Figure 2-9 (b) depicted that even though the two images are under similar class, they are not closer on the original space. Thus, this hash function puts these images in different bucket. To improve this weakness, metric learning algorithm (LMNN) is used in the hash function (Figure 2-9 (c)), it adds the learned constraints, so that images constrained to be similar will have high probability of hashing together in the same bucket. The circular green region in Figure 2-9 (b) showed that LSH function using random hyperplane on the original space generates a hyperplane uniformly at a random position to separate images. On the contrary, as in Figure 2-9 (c) illustrated, the red "Barn Red" region signified that our hash function bias the selection of random hyperplane to reflect the specified similarity and dissimilarity constraints. Therefore, metric learning algorithm improves the performance of Cosine similarity based hash function over original hash function overwhelmingly. It is one of the major contributions of this thesis, to use LMNN and ensure this performance improvement. Moreover, we claim that this work is the first to use LMNN algorithm in such context to learn Cosine randomized hashing. The experimental result is demonstrated under result and discussion Chapter.

Note: - Prateek Jain et al. [32] presented that there is tradeoff in the selection of bits number. Larger values of bit will increase the accuracy of how well the keys themselves

reflect the learned metric (Not all the time), but will increase the computation time and can lead to few collisions in the hash tables. On the other hand, lower values of *b* makes hashing faster, but the key will only coarsely reflect our metric, and too many collisions may result. Therefore, it is one of the weaknesses.

There are lots of recently demonstrated reasonable results for several large-scale benchmarks using this technique [38]. It allows representing of the intrinsic structure of the data in compacted binary format using hashing. Therefore, it is storage efficient. On the contrary, this leads to a drastically reduced evaluation effort [38]. On top of this, the hyperplane random line is independent from the data (without statistical information of the dataset). To ride off from the above limitations, we use another metric space hash function based on *p-stable* distribution (Euclidean metric space), is efficient [18].

2.7.2.2 Hash Functions based on p-stable Distribution

Like other metric space hashing function, *p-stable* distribution hashing is also carried out by mapping similar objects to the same bucket with higher probability than non-similar points using L_1 and L_2 space: Datar et al. [18] proposed hashing function in L_p norm and projects high dimensional data to *p-stable* random vectors as hash functions. Euclidean (L_2) space p-distribution LSH is known as *Exact Euclidean LSH* (E2LSH), use E2LSH in the entire thesis. A *p-stable* hash function *h* is defined as:

$$h_{a,b}(x) = \left\lfloor \frac{a.x + b}{w} \right\rfloor$$

Where *a* is a *d*-dimensional vector with entries chosen independently from a *p-stable* distribution (using *Definition 6*), chose random line and partition into equi-width segments of appropriate size *w*, and *b* is a real number chosen randomly from range $[0, w]$. and *h* maps vector *x* to an integer. Finally, based on to which segment the vector project onto, assign hash value, then it is intuitively clear that this hash function will be locality preserving in the sense of described in the previous section. The optimal value of *w* is depends on the dataset, but it suggested that *w*=4 gives good result [18, 66]. Thus, we use *w* = 4 in our implementation and experiment.

Definition 6:- Stable Distribution is a distribution D over $\Re$ is called *p-stable*, if there exists $p \geq 0$ such that for any n real numbers $v_1, v_2, v_3, \dots, v_n$ independent and identically distributed variables $X_1, \dots, X_n$ with distribution D, the random varibale $\sum_i v_i X_i$ has the same distribution as the varibale $(\sum_i |v_i|^p)^{1/p} X$, where X is a random varibale with distribution D [18].

It is known that stable distributions exists for any $p \epsilon (0,2]$. in particular [67]:

- a *cauchy distribution* D_C, defined by the density function; $c(x) = \frac{1}{\Pi}\frac{1}{1+x^2}$, is 1-stable (for Manhattan metric space)
- a *Gaussian (normal) distribution* D_C, defined by the density function; $g(x) = \frac{1}{\sqrt{2\Pi}} e^{-x^2/2}$, is 2-stable (for Euclidean metric space)

In addition, for this special case of $p = 1,2$. It is possible to calculate the probablities using the aforementioned density functions [18]:

- $p2 = 2\frac{tan^{-1}(w/c)}{\pi} - \frac{1}{\pi(w/c)}\ln(1 + (w/c)^2)$, $p = 1$ *cauchy and* where $c = 1 + \varepsilon$, ε is the approximate error,
- $p2 = 1 - 2norm\,(-w/c) - \frac{2}{\sqrt{2\pi}w/c}\left(1 - e^{-(w^2/2c^2)}\right) for\ p = 2$ G*aussian* where *norm(.)* is the cummulative distribution function for a random variable that is distributed as $\mathcal{N}(0,1)$. $the\ value\ of\ p1$ can be obtained by substituting *c = 1* in the formula above

The LSH scheme uses *Definition 6* to compute the dot product $(a.x)$ to assign a hash value for each vector x. Formally, each hash function $h_{a,b}(x): \Re^d \rightarrow \mathbb{Z}$ maps a d dimensional vector x onto the set of integers. Thus, each hash function is indexed by a choice of random *a* and *b*.

Other important parameters are: *k* (number of hash function) and *L* (number of hash table).

$$k = \log d/p2\ , \quad L = d^{\frac{\ln 1/p1}{\ln 1/p2}}$$

For detail mathematical formulation, refer the original paper and user manual of this algorithm [18, 66].

Even though Euclidean space LSH requires many hash table and consumes memory heavily (data points might have more than one hash value), it is very efficient [68]. Thus, we use this hashing to get its efficiency advantage. To achieve fast online query and less memory consumption, concatenation of all the hashing values to represent by a single value is another approach [68]. However, it reduces probability nearest neighbor point's collision and affects the performance as well. In order to guarantee the accuracy, *L* hash tables are used for each point to probe nearest neighbors in *L* buckets.

Generally, *p-stable* distribution based hashing functions corresponds to lines. Initially, partition the line into equi-width (or projection radius) segments of appropriate size *w*. The following step is to hash each point to the bucket containing its projection onto the line. As described above, nearby points have higher probability than distant points to be in same bucket. One of the contributions of this thesis is to enhance the performance of this hashing technique using metric learning algorithm (LMNN). We use LMNN to learn distance measure before plugging into the hashing function. The detail is explained in the next Chapter. The following figure shows how E2LSH is working.

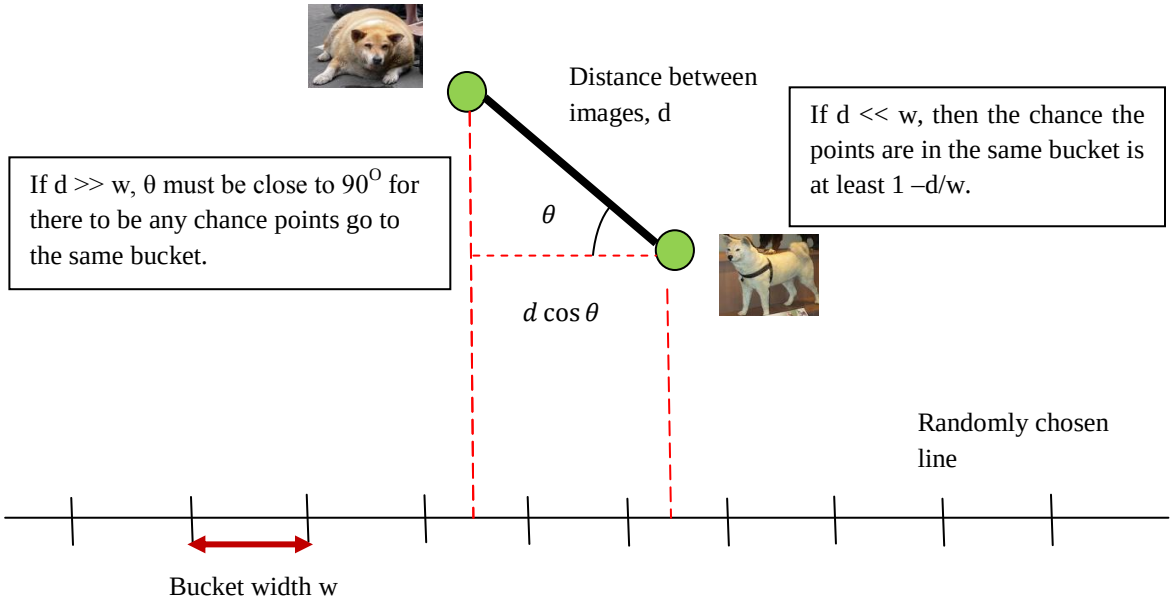


Figure 2-10:LSH for L_2 metric distance

To illustrate the above figure, in Euclidean distance hashing function family [69], if the distance between two points greater than *2w*, the angle is: $60 < \theta < 90$, there to be a chance

of these points going in the same bucket at most of *1/3* probability. On the other hand, if the distance is less than w/2, then the least chance to share the same bucket is *1/2*. Therefore, this hashing function yields a *(w/2, 2w, 1/2, 1/3)*-sensitive family of hash function.

To point up the contribution of this thesis, we use metric learning algorithm to keep data points under the same class closer than dissimilar classes by learning the metric distance a head of using the LSH. As a result, it enhances the efficiency of *p-stable* based distribution hashing technique, the experimental results is presented under result and discussion Chapter.

2.8 Summary

As discussed in the aforementioned sections, efficient way of addressing similarity search has been getting much attention in the area of image classification for the past decade, yielding a variety of tree and hash based techniques. However, traditional methods fail due to technical demands like; high dimensional data, specialized distance metric, and availability of supervision. In this research, we are interested to address these technical demands by breeding dimension reduction, metric learning algorithm, and hashing technique. This particular Chapter is dedicated to present the basic concepts and previous works of the aforementioned topics. The following specific topics are addressed: theoretical backgrounds of the research area, mathematical foundations to understand the mathematical formulation of metric learning algorithms, and also hashing (LSH) technique which is used to index data from database. In addition, all metric learning algorithms which are driven by nearest neighbor approach are summarized. Furthermore, locality sensitive hashing technique under Cosine similarity and Euclidean metric space is presented. Generally, Cosine similarity hashing is storage efficient; conversely, *p-stable* is not memory efficient.

Chapter 3

3.1 Methodology

This Chapter is dedicated to present different evaluation techniques which are used to evaluate our contributions versus similar algorithms to show the performance difference.

3.2 General flow

In this project, our input is labeled images. It is supervised classification problem. kNN classification is one of the most common and simple algorithm uses for such classification problem. Thus, we use it in this thesis. As shown in Figure 3-1, initially we normalize input data (X) to avoid the domination of one dimension over the others by giving similar scale. We use either Z-score or MinMax normalization depends the dataset [70],

$Normalized(x_i) = \frac{x_i - \mu}{\sigma}$, and $x_{i,new} = \frac{x_i - x_{min}}{x_{max} - x_{min}}$; where: $x_i \in X$ is the ith data point, x_{min} and x_{max} is the minimum and maximum respectively, μ the mean and σ standard deviation

This normalization step is followed by dimension reduction in order to reduce the large dimensionality of the input dataset. We use the most dominant dimension reduction technique called *Principle Component Analysis* (PCA). This technique helps to reduce the dimension without much loss of information. Thus, it helps to escape from the problem of the curse of dimensionality, and make fast the execution time.

The output of the above steps is the normalized and dimensionally reduced training data which is ready to be plugged into metric learning algorithm to train the data. In this thesis, out of the nearest neighbor driven approach metric learning algorithms, we use large margin nearest neighbor (LMNN) [56]. It performs very well in practice compare to other

related algorithms. Its main objective is to minimize the distance between similar classes and maximize the distance for dissimilar classes of training examples. The theoretical reasons are mentioned in the previous Chapter to answer why we chose this algorithm from other related algorithms like MMC [8], NCA [5], and MCML [55]. In addition, the next Chapter will justify with experimental demonstrations.

The output of this metric learning algorithm is used to project the data into another space which visibly keep data points of similar class closely, and large margin distance for dissimilar classes ("*impostors*").

As illustrated in the Figure 3-1, likewise the original paper [56], we have evaluated the performance of this algorithm and compare with Euclidean metric distance. The experimental result is presented in the next Chapter. As shown in the Figure 3-4, to learn the indexing technique, the output of this metric learning algorithm is used as an input to LSH process. As a result, it learns the indexing technique.

The most common techniques to asses accuracy of the classifier based on randomly sampled partitions of given data are holdout, cross-validation, and bootstrap [70]. These techniques are presented as follow [70]:

- In the holdout method, input data partition randomly into two independent sets, a training and test set. Typically, two thirds of the data are allocated to the training set, and the remaining is allocated to the test set [70]. The training set is used to build model, whose accuracy is estimated with the test set. The main drawback of single iteration holdout is that the samples might not be representatives. Thus, random sub-sampling is a type of holdout method in which holdout method is repeated *k* times. The accuracy is estimated by computing the average of accuracies obtained from all iteration.

- In cross validation, unlike holdout method; the initial step is to partition the data randomly into *k* mutually exclusive subsets or folds; F_1, $F_{2,}$.....,F_k, each of approximately equal size. Training and testing is performed *k* times. In the iteration *i*, partition F_i is reserved as testing set, and the remaining as training set. This method is called *Leave-one-out*. The main drawback of this technique is computationally very expensive. Due to lack of time, we do not use this technique.

- Bootstrap is a cross validation uses sampling without replacement; 63.2% of the original data tuples are under training set, and the remaining 36.8% will form test set (hence, the name, .632 bootstrap) selected once. It is a best technique for very small datasets.

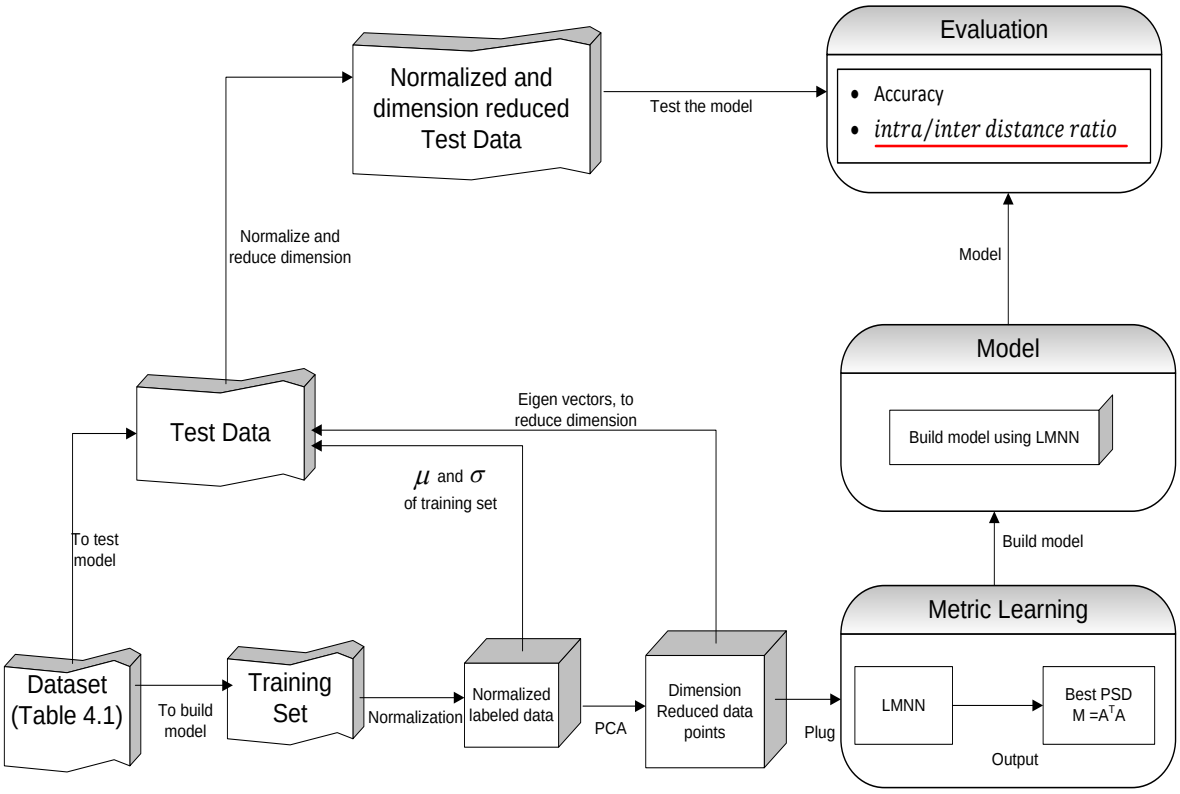


Figure 3-1: Model of metric learning algorithm building and evaluate performance

As presented in the previous Chapter, the main weakness of LMNN is to set prior the optimal number of target. To resolve this we use cross-validation method on the training set for further partition, training and validation set.

Generally, out of the aforementioned common techniques for assessing accuracy, both holdout and cross-validation are implemented in this thesis.

In depth, the following sections answer what and how the evaluation techniques are used.

3.3 Metric learning evaluation

As shown in the above figure, the data is initially classified as training and testing set. The training set is passed through normalization and dimension reduction before plugging into metric learning to build the model. On the other hand, we use test set to evaluate the performance of the model. We use the output of normalization and dimension reduction step of training set to affect the test set. Finally, test set is ready to evaluate performance of the model. The following two techniques are used over the holdout randomly sampled partitions of a given dataset.

i. Accuracy

The accuracy of the classifier is used to calculate the percentage of the test set examples that are correctly classified by the classifier. We compare the kNN classification accuracy and error rates ($\{accuracy_{rate} + error_{rate} = 1\}$) of Mahalanobis versus Euclidean metric distance. To break the tie among different classes, we use two different techniques either to take the *most nearest* or *random selection* from *k* points. For example, if the problem is 5NN (refer Figure 3-2). If two of them are class "Yellow", the other two are class "Red", and rest one is class "Green". In this circumstance, there is a tie between "Yellow" and "Red" class. To resolve the tie, we take either the most near or chose random selection from the four nearest data points (both Yellow and Red classes) to predict the class of the test example. We use *random selection* to break the tie for all experiments.

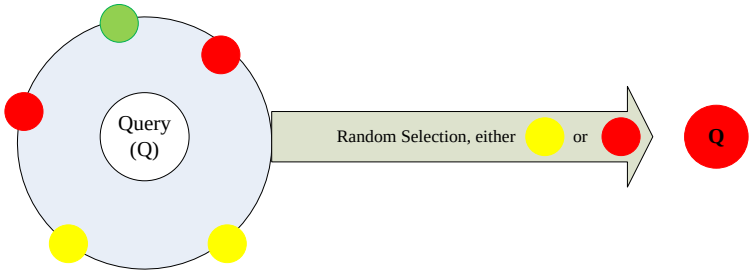


Figure 3-2: kNN tie resolution,random selection

ii. Intra to inter distance ratio

Based on the theoretical ground of LMNN, makes the *intra-distance* (distance between similar classes) much smaller than the *inter-distance* (distance between dissimilar classes). In the context of this nature, we contribute the following evaluation technique: calculate the average intra to inter distance ratio for both Euclidean and Mahalanobis metric distance, and compare the ratio. Based on this theoretical ground; this hypothesis is designed:

$$Mahanalobis\ [intra\ /\ inter]\ ratio\ is\ smaller\ than\ Euclidean.$$

The experimental result of this average distance ratio comparison is presented in the next Chapter. The following section will present how we evaluate the performance of the hashing techniques.

3.4 Hashing Evaluation

To evaluate the performance of LSH implementation, there is no public database containing query points. Thus, we construct our own query set from the entire dataset. We use simple holdout random sampling technique to split the whole dataset into two disjoint set. The first split is used to build the hash table (training set) which contains 90% and the remaining 10% is used as a query set (testing set). Figure 3-4 shows the general flow of the thesis.

We evaluate our solution using two different techniques which are:

I. *Computational Complexity*

As discussed in the previous Chapter, there are steps which are taking place in the building and testing performance of hash table. These are: projection (either with metric learning or not, offline), hash each data points using the hashing function, generate signature for each data points (from hashing function), preserve all data points which has similar signature in the same bucket else different, and the last step is to test the performance by searching the nearest neighbor for a given query point. In this thesis, we use *approximate nearest neighbor* (ANN) search technique which explained in the previous Chapter. These steps are used in this thesis to compare the computational complexities of our solutions with generic hashing and exhaustive techniques. The summary of this computation complexity is presented in the next Chapter.

II. *Query accuracy,*

The initial step is to access bucket(s) which has a radius of *r* from a given query point, ANN search. The next step is to apply the generic kNN classification accuracy rate procedure. The following figure illustrates the method.

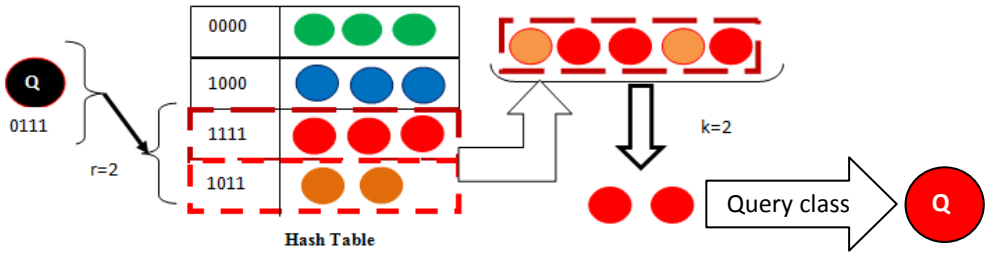


Figure 3-3: Schematic illustration of query accuracy, r=2, 2NN

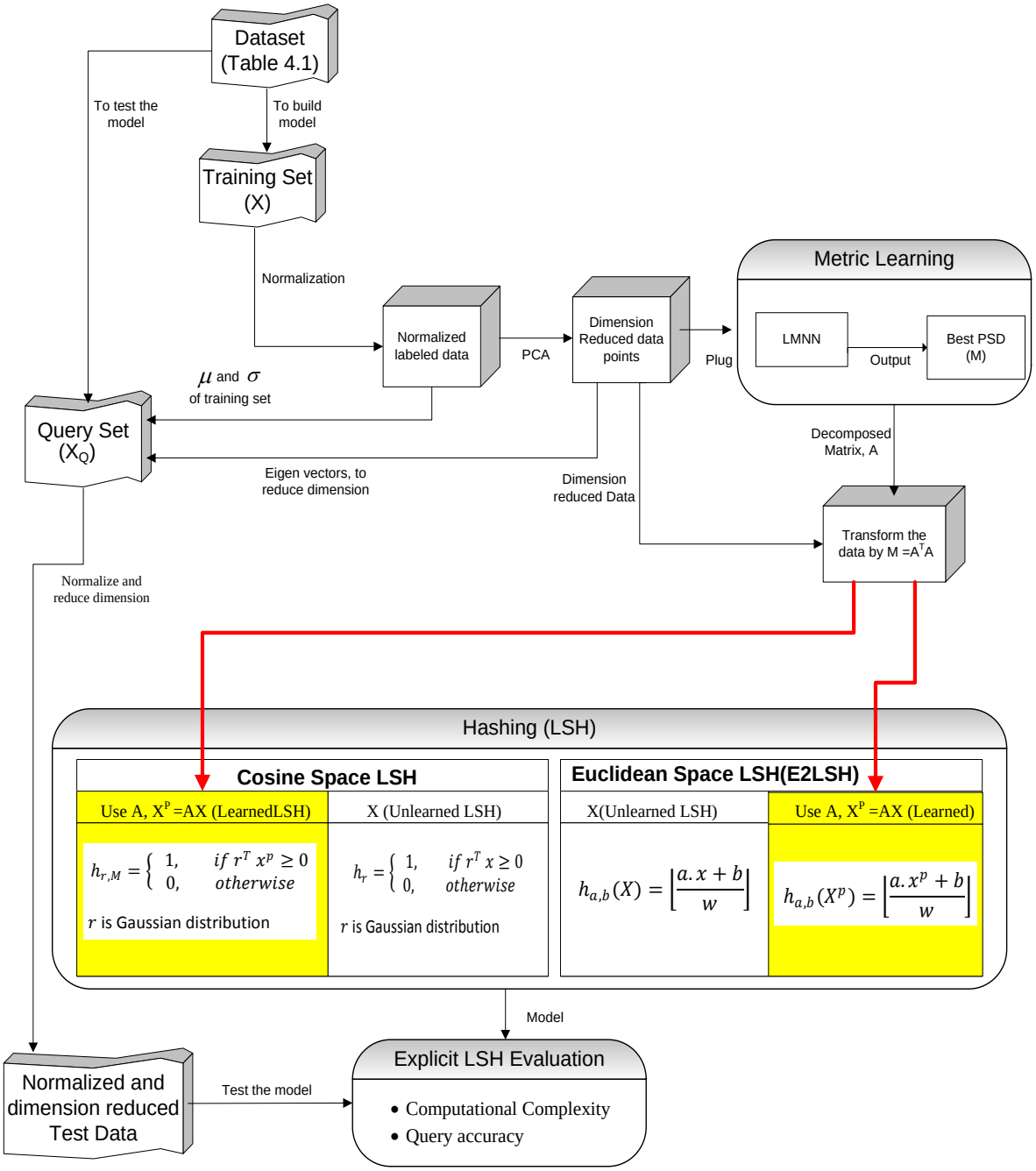


Figure 3-4: Model of building learned hashing techniques and evaluate performance

The following section will present the platform, libraries and other resources used to implement all the algorithms and evaluation techniques.

3.5 Platform to Implement

According to our experience, the most common platforms to implement machine learning algorithms are Matlab and Python. As a reason it is very easy to get help. However, in this project due to the requirement of the company (EURA NOVA)[5], all the algorithms and evaluation techniques are implemented in Java JDK 1.7(Java SE programming language) under Eclipse Integrated Development Environment (IDE).

Naturally, Java does not support matrix and some algebraic computations. To carry out these computations, it has to be implemented using built-in data structures like array. However, this implementation is tough and time consuming. Therefore, we use other external libraries to get all the necessary algebraic computations. These libraries are listed in the following table with their purpose and also link to browse.

Table 3-1: List of libraries

Library, (.jar)	***Version***	***Purpose***	***Link***
combinatoricslib	2.0	To generate combinatorial objects	*code.google.com/p/combinatoricslib/*
commons-lang	3-3.3.2	Basic numerical methods	*commons.apache.org/lang/*
commons-math	3-3.2	For mathimatical, and statistics componenet	*http://commons.apache.org/proper/commons-math/*
Jama	1.0.3	Basic linear algebra package for java	*http://math.nist.gov/javanumerics/jama/*
Javaml	0.1.5	Collection of machine learning algorithms	*http://java-ml.sourceforge.net/*
Six11utils	----	Includes networking, math, IO, GUI, and logging classes	*https://code.google.com/p/six11utils/*
Ujmp-complete	0.2.5	For sparse and dense matrix, linear algebra, visualization, big data	*http://sourceforge.net/projects/ujmp/files/ujmp-complete/*
Utils	----	Matrix manipulation	*http://www.seas.upenn.edu/~eeaton/software/Utils/javadoc/*
Jsat	----	A Java GUI and library to quickly analyze, model, and predict data	*https://code.google.com/p/java-statistical-analysis-tool/downloads/list*

To evaluate the performance of our algorithms, we use EURA NOVA[4] server computer. To manage and build our implementation, Apache Maven[6] is used.

[5] http://euranova.eu/
[6] http://maven.apache.org/

Chapter 4

4.1 Result and Discussion

This chapter of the thesis is dedicated to present experimental results of both metric learning algorithms and hashing techniques. In the metric learning algorithms experiment, we compare other metric distance with LMNN to justify why we propose to use LMNN. Moreover, the experimental evaluations of hashing techniques are also presented to show the effect of metric learning in LSH. Mainly, this chapter is summarized into three major subsections. The first section is to present the experimental results of metric learning algorithms, followed by hashing solutions. The last section is discussion to summarize all the experimental results. For these evaluations, we used different datasets from the UCI machine learning repository[7], New York University Computer Science (Data for MATLAB hackers)[8] and deep learning[9].

4.2 Datasets

We use various datasets from the above section mentioned sources. These datasets are listed in the following table. These datasets are derived from collections of images, and text. Before we plug the dataset into our distance learning algorithm; both normalization (*z-score* and *min-max* normalization) to avoid bias, and dimension reduction (using PCA) to make the execution fast take place. The detail flow of the thesis is presented in the previous Chapter. Thus, the following table shows the statistics of the datasets which are used in our experiments; including their actual and reduced dimensions.

[7] https://archive.ics.uci.edu/ml/datasets.html
[8] http://www.cs.nyu.edu/~roweis/data.html
[9] http://www.cs.utoronto.ca/~kriz/cifar.html

Table 4-1: List of datasets

Dataset	***Input***	***Actual Dimensions***	***Reduced Dimension***	***Class***	***Source***
Wine	178	13	13	3	UCI
Iris	150	4	4	3	UCI
Balance-Scale	535	4	4	3	UCI
Letter-recognition	20000	17	17	26	UCI
OlivettiFaces	400	4096	200	40	Data for MATLAB hackers
isolet	7797	617	200	26	UCI
Mnist	70000	784	200	10	Data for MATLAB hackers
Cifer-100	50000	3072	200	100	Deep Learning

Using these reduced dimension datasets, we measure the performance of LMNN algorithm and hashing techniques. Prior of presenting the experimental result, the setup of the experiment is presented, and followed by the experimental results to justify our theoretical hypothesis.

4.3 Setup

The primary general objective of LMNN classification is to improve the kNN classification using Mahalanobis metric distance. The learning process is led by semi-definite programming. Computationally, it is very expensive especially when we have large dimension. Thus, we use dimension reduction technique. Moreover, LMNN algorithm works better with PCA to reduce dimension of some datasets in order to speed up the process of learning and avoid overfitting [7]. Therefore, we use PCA to reduce dimension of a given dataset before plugging into LMNN learning process.

One of the most crucial variables for optimization of LMNN classification is a target neighbor has to be assign prior the learning process. To calculate this value, we use Euclidean metric distance in the input space (on reduced dimension) using cross-validation on training set. This technique is called *multi-pass LMNN*. Due to lack of time, likewise, the original paper of LMNN, for all experiments, we set constant for the number of target, *target neighbors* ($NoTarget = 3$) [7].

Furthermore, in our metric learning kNN classification accuracy rate (or error rate) experiments, the value of *k (number of nearest neighbor)* is the most important input. We set constant for the entire experiment, *k=5*.

To justify the theoretical ground of LMNN and why we choose this algorithm; these evaluation techniques are used: *intra/inter* average distance ratio and kNN classification accuracy rate; and compare with other distance metric. We obtain the experimental results for metric distance is by averaging multiple runs (10 and 3 times for small and large size datasets respectively) of randomly generated 70/30 split of each dataset under training and testing set respectively. In kNN classification accuracy and error rate performance evaluation, to break *ties* among different classes; *random selection* technique is used which is explained in the methodology Chapter. In addition, we use the experimental results of the LMNN original paper [7] to show how this algorithm is performing better than other related algorithms. The other important parameters for LMNN implementation are listed in the following table including their value used in the experiment.

Table 4-2: LMNN parameters

Parameters	***Details and declaration***
Min_iter	Minimum iteration, 1000
Max_iter	Maximum iteration, 10000
η	Learning rate, 0.1
μ	Weighting of pull and push terms, 0.5
Tol	Tolerance for convergence, 0.00001
Best_C	Best error obtained so far, its initially value: double.MAX_VALUE
Best_M	Best Metric so far, initially value: Identity Matrix

To evaluate the performance of hashing implementation, the dataset is randomly split into two disjoint set, 90/10 of training and query (or test) set. There are two technique we use to evaluate the performance; computation complexity, and query accuracy. The detail is explained in the previous Chapter. The parameters used in the implementation of the hashing techniques (Cosine similarity and E2LSH) are listed in the following tables

including their explanation with their value used in the entire experiment. To assign values for these parameters, we reviewed papers and did preliminary experimental tests.

Table 4-3: Cosine LSH parameter

Parameters	***Details and declaration***
r	Random *r*, Gaussian distribution with zero mean and unit variance
$radius = (1 + \varepsilon)$	To approximate NN, guarantee retrieval of examples within the radius, 2

Table 4-4: E2LSH parameter

Parameters	***Details and declaration***
$c = Radius(\varepsilon + 1)$	ε approaximation error, c multiplier used to control the degree of approximation > 1, $\varepsilon = .5, c = 1 + .5 = 1.5$
w	Projection radius, 4
k	Number of hash functions, depends on the dataset's dimension
l	Number of separate hash tables, depends on the dataset's dimension
b	Real numbers choosen from range of [0,w]
radius	Search NN in this *radius*; 2.5

To answer the research question, we apply metric learning algorithm in the existed hashing technique and contribute a fresh automatic large-scale image classification approach. To evaluate the performance of our solutions', query accuracy test is used and compare with the existed traditional hashing, and exhaustive techniques. Furthermore, computational complexity of the main steps of hashing techniques is used also. In this experiment, we use the formula below to evaluate the average query accuracy rate. The value of *k* is constant (*k* = 10), and as mentioned above the query size is 10% of the entire dataset.

$$\text{Avg. Query\_Accuracy}_{\text{LSH}} = \frac{1}{\#k}\left(\sum_{k=1}^{\#k}\frac{\sum_{i=1}^{\#querySize} k\text{NNaccuracyRate}}{\#querySize}\right)$$

Finally, the impact of metric learning in the hashing, and its performance of classification is summarized in the last discussion section.

4.4 Metric learning Experiment

In this experiment we use some of the datasets which are listed in the Table 4-1, to present experimental results using both *intra/inter* average distance ratio, and accuracy rate evaluation techniques. The experimental results are explained in the following subsections.

4.4.1 Intra/inter ratio

As presented in the second Chapter, the main theoretical ground of LMNN algorithm is to minimize the distance between data points which has similar class than dissimilar class. To justify this objective function, it is convenient to use the average distance ratio between similar and dissimilar class of data points (*intra/inter* average distance ratio).

Based on this objective function, we declare the following hypothesis:

Hypothesis 1: *intra to inter class average distance ratio of LMNN is smaller than Euclidean distance*.

For this evaluation technique, three different datasets (wine, balance scale, and mnist) are used from the listed datasets, Table 4-1. The following figures are depicted the experimental results of *intra/inter* class average distance ratio of LMNN Vs Euclidean metric distance.

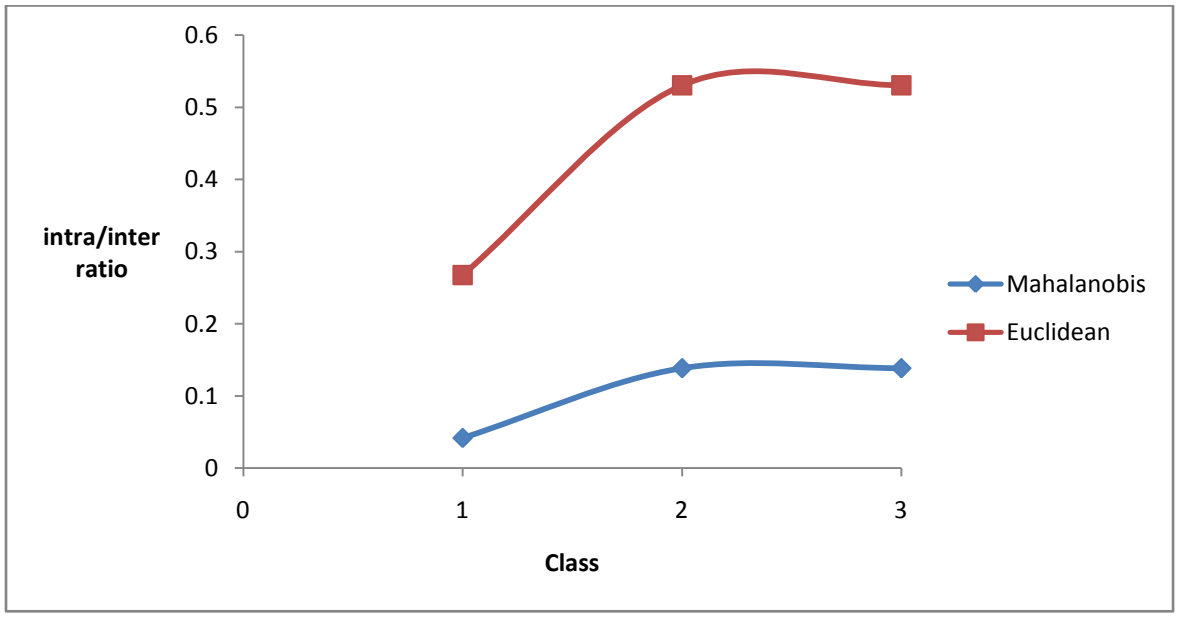


Figure 4-1: Wine, intra/inter distance ratio

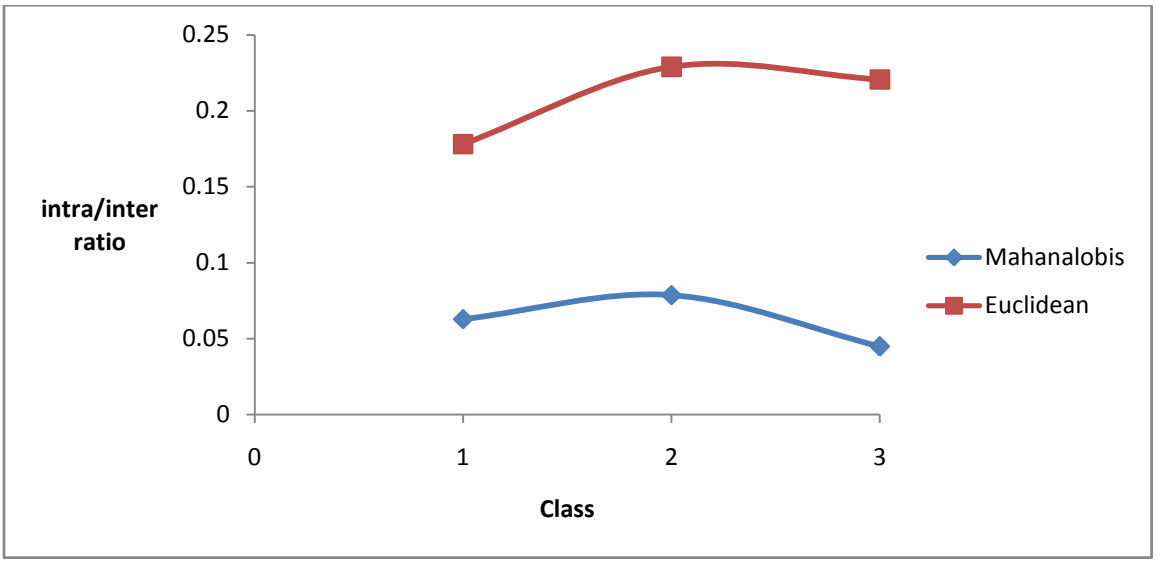


Figure 4-2: Balance-Scale, intra/inter ratio

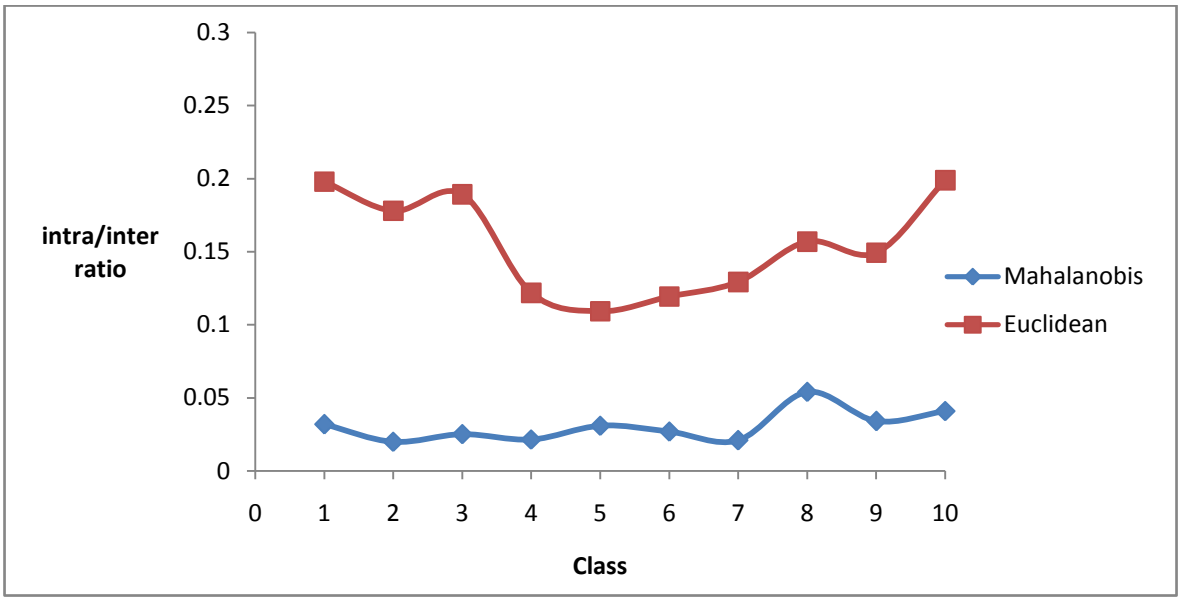


Figure 4-3: Mnist, intra/inter ratio

The above three experimental result figures depicted that the average distance ratio of intra to inter (*intra/inter*) class of LMNN is significantly smaller than Euclidean distance metric. Therefore, the effect of pushing and pulling between dissimilar and similar classes respectively; affects the distance between points enormously. As a result, it minimizes distance between data points which are sharing class, while maximize distance between different class. Due to the effect of LMNN algorithm, intra-distance is much smaller than

inter-distance. Therefore, it is expected that LMNN has competence of maximizing kNN classification accuracy rate. To validate this premise the following section is designed.

4.4.2 Accuracy Rate

Based on the *intra/inter* average distance ratio's experimental results which are presented in the above section, the following hypothesis is asserted:

Hypothesis 2: *LMNN has higher kNN classification accuracy and lower error rate than Euclidean metric distance. It means: LMNN improves the traditional kNN classification performance.*

To testify the above hypothesis, kNN classification accuracy and error rate are used to evaluate the performance of LMNN, and compare with other metric distance.

The following figure (refer Figure 4-4) shows the kNN classification accuracy rate experimental results of wine dataset for ten different values of k ($1 \leq k \leq 10$).

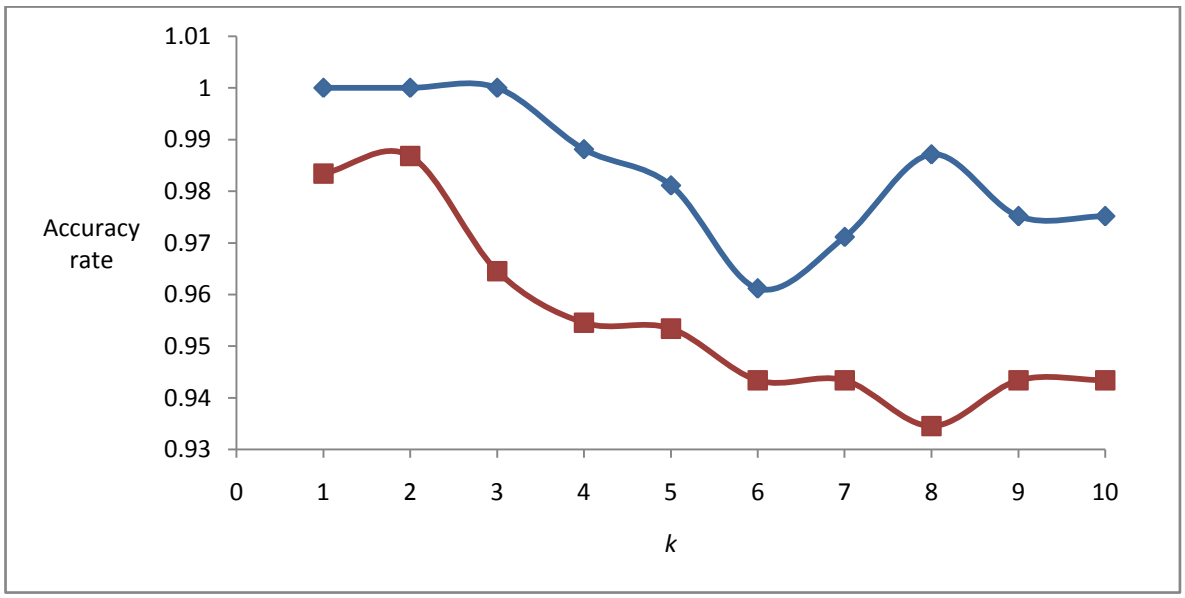


Figure 4-4: Wine, Accuracy rate LMNN Vs Euclidean Metrics, ($1 \leq k \leq 10$)

The above Figure 4-4 clearly shows that LMNN outperforms Euclidean metric distance in all the ten values of *k*. To justify this hypothesis with more and large datasets, another experiments take place using kNN classification error rate by assigning constant for the value of *k* (*k=5)* which is shown in the Figure 4.5.

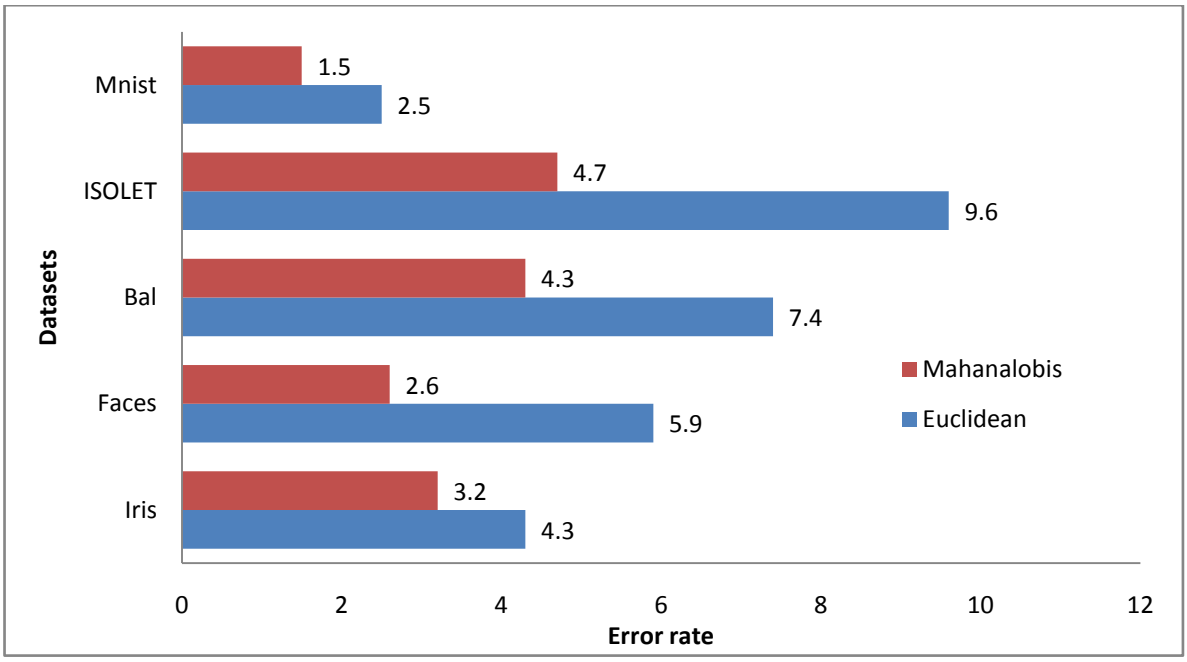


Figure 4-5: Error rate LMNN Vs Euclidean Metrics, (*k =5*)

The above Figure 4-5 clearly shows that LMNN kNN classification error rate is smaller than Euclidean metric space. In other word, LMNN improves the original kNN accuracy rate significantly. Even though the computation of LMNN is expensive, it is worthy to use if accuracy is the main objective. The main objective of this thesis is classification of large-scale image database in which accuracy is the most important. Therefore, LMNN is suitable to answer the objective of the thesis.

In addition to the above presented experimental results, we use the original LMNN paper's kNN error rate's experimental evaluation results and comparison with other related algorithms result; to support our justification. The comparison takes place with RCA (Relevant Component Analysis), MMC, PCA, and NCA [10]. To break a *tie*, they use a technique of repeatedly reducing the size of neighborhood. As per the paper, both the MMC and NCA implementation ran out of the memory for large datasets as shown in the table below, label as *N/A*. As per the objective of the thesis, inefficiency of processing large datasets is adequate reason not to consider it. The following Table 4-5 summarizes the results of kNN classification error rate using the aforementioned distance metric using both small and large datasets.

Table 4-5: kNN classification error rate Experimental results LMNN Vs other related metric space [10]

Statistics	***Mnist***	***Letters***	***Isolet***	***Bal***	***Wines***	***Iris***
#inputs	70000	20000	7797	535	152	128
#features	784	16	617	4	13	4
#reduced dimensions	164	16	172	4	13	4
#training examples	60000	14000	6238	375	106	90
#testing examples	10000	6000	1559	161	46	38
#classes	10	26	26	3	3	3
kNN						
Euclidean	2,12	4.68	8.98	18.33	25.00	4.87
PCA	2.43	4.68	8.60	18.33	25.00	4.87
RCA	5.93	4.34	5.71	12.31	**2.28**	**3.71**
MMC	N/A	N/A	N/A	15.66	30.96	3.55
NCA	N/A	N/A	N/A	**5.33**	28.67	4.32
LMNN						
PCA	1.72	3.60	4.36	11.16	8.72	4.37
Multiple Passes	**1.69**	**2.80**	**4.30**	5.86	7.59	4.26

As shown in the above Table, on the first three small datasets (wine, walance-scale, and iris), LMNN classification improves kNN classification accuracy rate compare with traditional kNN classification using Euclidean distance metric in both before and after affecting by PCA. On the other hand, compare to the other metric learning algorithms: in addition to the aforementioned limitation of MMC, outperforms by LMNN. On the other hand, even though LMNN performs better in these datasets, compare with NCA and RCA the results got somewhat variable. In some case LMNN performs better, but worse in others. However, the size of these datasets does not represent the input dataset for this project. For this reason, three large datasets are used as shown in the above table. The experimental results show that LMNN outperform all the above mentioned metric learning algorithms significantly. Therefore, LMNN algorithm is suitable to use.

4.5 Hashing Experiments

As discussed in the previous Chapters, to achieve the objectives of this thesis, LSH technique is simple and intelligent solution. Out of all the domain of different metric space hashing techniques, both Cosine similarity and *p-stable* (Euclidean space, E2LSH) hashing are used in this thesis. To improve the performance of these generic hashing techniques, metric learning (LMNN) is used to learn metric distance, and gain its benefit and improve the hashing performance. Mainly, this is the main contribution of this thesis. To evaluate the performance of our solution, the datasets which are listed in Table 4.1 are used.

Two evaluation techniques are used to assess the performance of our solutions and compare with generic hashing and exhaustive techniques. These evaluation techniques are: computational complexity, and kNN classification of query accuracy rate. These methods are presented in the previous Chapter.

4.5.1 Computational Complexity

As presented in the entire thesis, the main contribution of the project is to construct hash table by breeding both metric learning algorithm and hashing techniques under different metric space in order to classify large-scale image database. Under this construction, there are major steps take place in any metric space hashing: projection of the dataset, decide hash functions and number of signature (*b*, bits in Cosine similarity LSH) to represent a given data point, and compute the hash functions. Finally, it is ready to access the nearest neighbor for a given query point. To access the ANN for a given query point, there are existing methods [16, 64]. The theoretical background of this ANN is presented in the second Chapter. Given *N* data points has unique signature(s) using either Cosine similarity hashing or E2LSH. In this thesis we employ the method of [64], which guarantees searching of $M = 2N^{1/(1+\varepsilon)}$ data points to get the *k* nearest neighbors. The number of ANN is much smaller if we have very large dataset ($M<<N$). The following table summarizes the computational complexity of the aforementioned steps of the algorithms and also exhaustive search methods to compare their complexity.

Table 4-6: Computational complexity [32 18]

Steps	***CosineLSH***	***CosineLSH +LMNN***	***E2LSH***	***LMNN + E2LSH***	***Euclidean***	***LMNN***
Metric learning projection (offline)	$O(d)$	$O(d^2)$	$O(d)$	$O(d^2)$	$O(d)$	$O(d^2)$
Hash functions	$O(b)$	$O(b)$	$O(lk)$	$O(lk)$	$O(0)$	$O(0)$
Signature (to represent the data point)	$O(1)$	$O(1)$	$O(l)$	$O(l)$	$O(1)$	$O(1)$
Hashing: compute	$O(bd)$	$O(bd)$	$O(dlk)$	$O(dlk)$	$O(0)$	$O(0)$
Search: identity the query's ANNs	$O(Md)$	$O(Md)$	$O(lMd)$	$O(lMd)$	$O(dN)$	$O(dN)$

As shown in the above Table, five main steps are used to evaluate the computational complexities of all the implemented algorithms and compare each other including exhaustive techniques (Euclidean and LMNN).

The initial step is to affect the dataset by the output of LMNN to gain the benefits of metric learning algorithm. In this step, all the algorithms which use the output of LMNN algorithm have higher complexities than others. When they use LMNN, each data points are projected into another space by the output of LMNN algorithm; else there is no projection. Whenever, there is projection the complexity is$O(d^2)$: else $O(d)$.

The second and third step is to decide the number of hash functions and represent each data point using unique signature(s) respectively. It is intuitive that exhaustive techniques have a unique signature. Cosine similarity hashing also uses a single unique sequence of bits (*b*) to represent a given data point. The key factor is to decide the number bit sequence, the number of hash functions which randomizes vectors. Therefore, each data point has a single signature, gets from *b* number of hash functions. In the case of *E2LSH*, the number of hash functions (*k*) and signature(s) (*l*) to represent a given data point is calculated by using the dimension of the

dataset and probability distribution. Each of *l* signatures is computed from *k* number of hash functions. Therefore, it is intuitive that both with and without metric learning *p-stable* distribution (Euclidean space) hashing uses heavy memory compare to the other methods.

The fourth step is the most important to build hash table. In this step, the E2LSH technique computational complexity is expensive. Similarly, if the value of *b* is large, Cosine hashing computation will increase. As mentioned above, each data points in E2LSH hashing are addressed by *l* number of keys. Due to this, this hashing uses heavy memory consumption. Nevertheless, it is very efficient. To justify its performance, we use query accuracy test which is presented in the next section.

After building the hash table, the final step is to evaluate the performance of hashing table using their computational complexity to access ANNs for a given query point. Both exhaustive methods (Euclidean and LMNN) are efficient for small data size, but it is intractable to use in our scenario. Compare to Cosine hashing; E2LSH increases the complexity of accessing ANNs by *l* hash table(s) factor.

4.5.2 Query Accuracy Rate

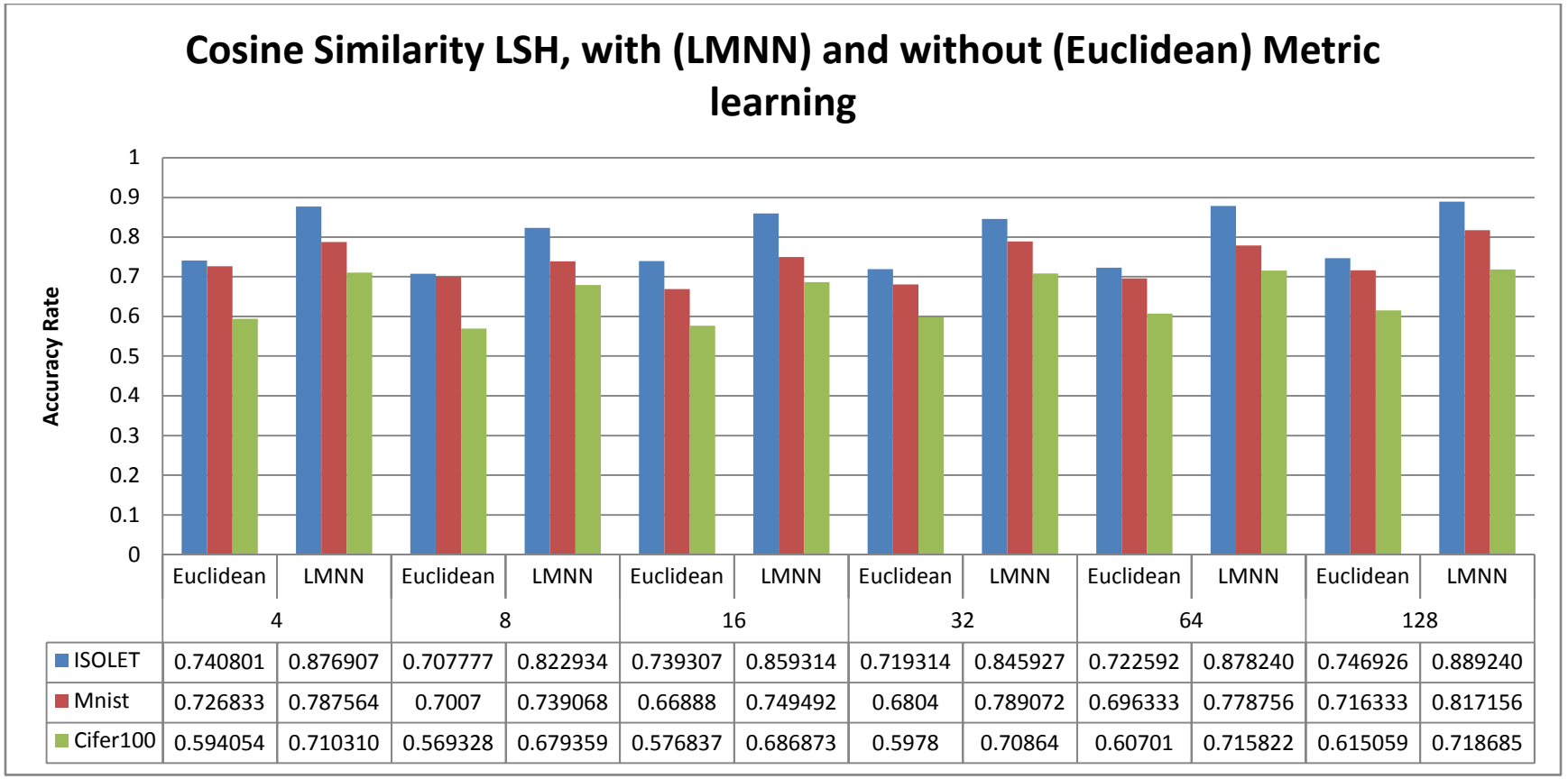
In this subsection, we use kNN classification accuracy rate of query points to evaluate the performance of our solution, and compare with generic hashing and exhaustive methods. As shown in the above section, in every case of using metric learning; the computational complexity is growing. Even though the complexity is growing, it improves the performance of classification. To justify this performance improvements of using metric learning algorithm (LMNN); this section is designed. Prior to presenting the experimental results, the above theoretical ground leads us to assert the following hypothesis:

Hypothesis 3: *Using LMNN algorithm in both Cosine similarity and Euclidean metric space LSH improves the accuracy rate*.

To justify this hypothesis, the experimental results for both Cosine similarity and Euclidean metric space hashing implementation are presented in the following subsection.

4.5.2.1 Cosine Similarity LSH Query Accuracy Rate

This subsection is dedicated to justify Hypothesis 3 by learning Cosine similarity LSH using LMNN to improve the classification accuracy. Three datasets (Isolet, Mnist and Cifer100) from Table 4.1 are used for this experiment. The experimental result is shown in the figure below:



	Euclidean	LMNN	Euclidean	LMNN	Euclidean	LMNN	Euclidean	LMNN	Euclidean	LMNN	Euclidean	LMNN
	4		8		16		32		64		128	
ISOLET	0.740801	0.876907	0.707777	0.822934	0.739307	0.859314	0.719314	0.845927	0.722592	0.878240	0.746926	0.889240
Mnist	0.726833	0.787564	0.7007	0.739068	0.66888	0.749492	0.6804	0.789072	0.696333	0.778756	0.716333	0.817156
Cifer100	0.594054	0.710310	0.569328	0.679359	0.576837	0.686873	0.5978	0.70864	0.60701	0.715822	0.615059	0.718685

Figure 4-6: Cosine similarity with Vs without LMNN LSH kNN accuracy rate experimental result summary

Figure 4-6 clearly shows that learned Cosine similarity hashing performs better than randomized hashing function due to the effect of metric learning algorithm, LMNN. LMNN plays role to learn the distance metric and improve the kNN classification. As a result, it changes the randomized hashing into learned hashing while improving the classification accuracy rate significantly.

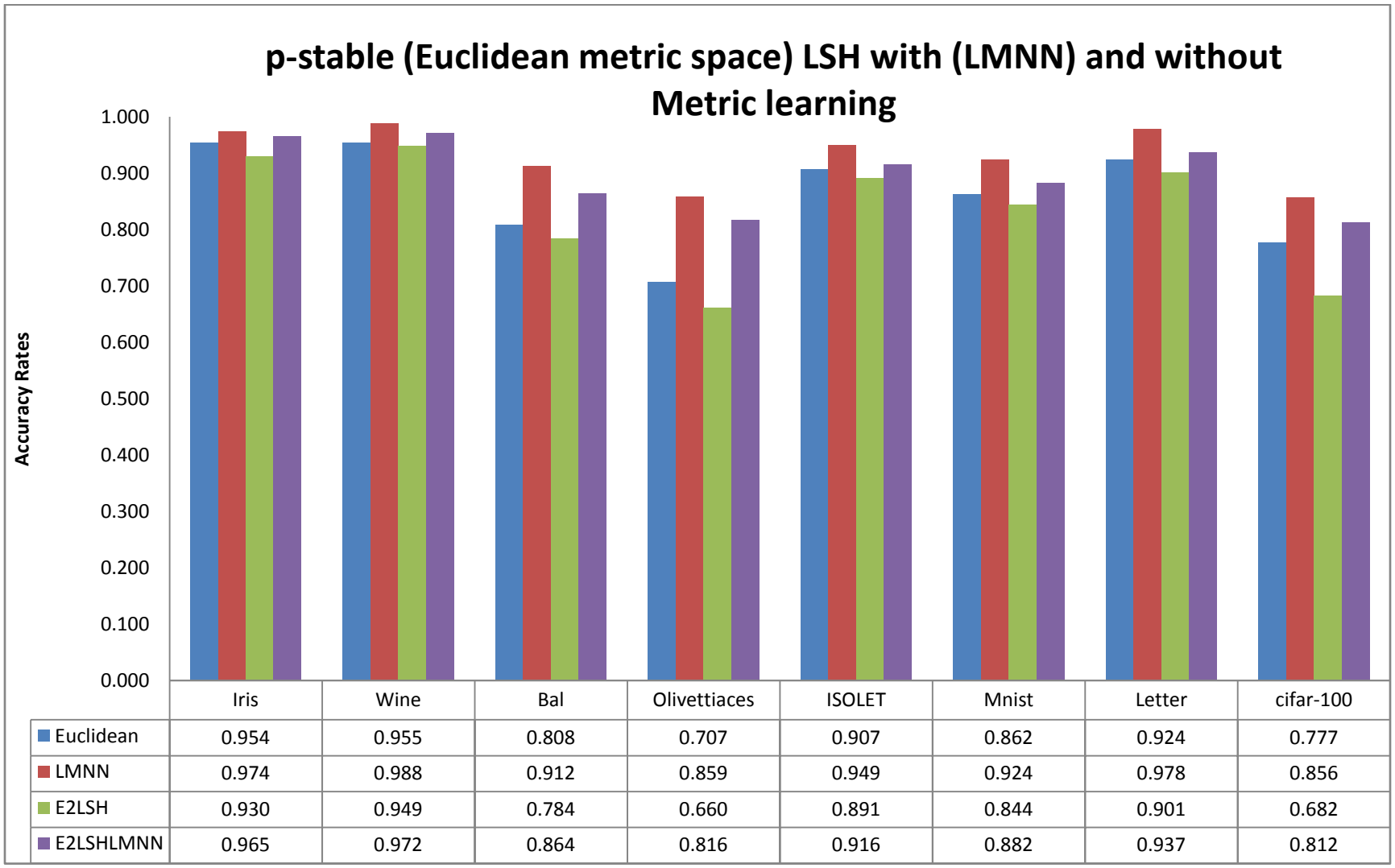
The generic randomized hashing technique uses randomized hyperplane to classify data points without considering the distribution of the data. Due to this, there are different approaches to improve the performance. Using metric learning is one of the approaches. Thus we use LMNN to improve its performance, is one of the contribution of this thesis. Even though it improves the performance, it has tradeoff in the choosing of number of bits and the hyperplane is random as well. Moreover, keep in mind the tradeoff, large number of bits increases the accuracy (but not linearly) and computational time grows (linear). To ride off from this problem and get a better performance, we implement another metric space hashing called *p-stable* distribution (Euclidean metric space, E2LSH) hashing technique.

4.5.2.2 p-stable Distribution (Euclidean Metric Space) LSH Query Accuracy Rate

Unlike Cosine similarity hashing, *p-stable* distribution uses l number of signatures to represent all the objects in the database. Literature shows that Euclidean LSH is efficient, but requires many hash tables and consumes memory heavily [68]. Thus, it is suitable hashing technique to answer limitations of Cosine similarity and to attain the objective of this thesis. To maximize the efficiency and accuracy of this algorithm, we deploy metric learning algorithm which is one of the contribution of this thesis. Moreover, we ensured that this metric learning algorithm has never used for such application before. To justify this premise, kNN classification query accuracy rate evaluation technique is used. The detail of this method is explained in the previous Chapter.

This experiment is designed to testify this hypothesis which is designed based on the above statements:

Hypothesis 4: *Due to the compact representation of Cosine Similarity LSH, Euclidean metric space LSH performs better.*



	Iris	Wine	Bal	Olivettiaces	ISOLET	Mnist	Letter	cifar-100
Euclidean	0.954	0.955	0.808	0.707	0.907	0.862	0.924	0.777
LMNN	0.974	0.988	0.912	0.859	0.949	0.924	0.978	0.856
E2LSH	0.930	0.949	0.784	0.660	0.891	0.844	0.901	0.682
E2LSHLMNN	0.965	0.972	0.864	0.816	0.916	0.882	0.937	0.812

Figure 4-7: p-stable Euclidean metric space with and without LMNN LSH, and exahustive methods kNN accuracy rate experimental result summary

As shown in the above experiment, from small to large dataset size LMNN linear search performs better than Euclidean metric distance, and Euclidean space hashing in both with and without using LMNN. To gain the benefits of LMNN, we plug LMNN into our Euclidean metric space hashing (LMNN + E2LSH). It improves the accuracy rate significantly compare to in the original space hashing, E2LSH. In addition, compare to learned Cosine similarity hashing, E2LSH with LMNN performs far better. As mentioned in the aforementioned paragraph, E2LSH does not concatenate *k* hash functions like Cosine similarity. As a result, Cosine similarity has fast online query for small than large number of bits. However, it has a tradeoff. On the other hand, due to the size our input dataset, exhaustive techniques are not feasible to use.

Generally, the main goal of this research is to devise classifier by breeding metric learning and hashing technique which improves the classification accuracy compare to the original method. Based on the aforementioned experimental results we put all the techniques in the following order, good to poor accuracy rate: LMNN, LMNN+E2LSH, Euclidean metric space, E2LSH, LMNN + Cosine LSH, and Cosine LSH. In all the cases, we get better performance whenever we use metric learning. Therefore, learning distance is very important and plays great role to improve the performance of both Cosine and *p-stable* distribution hashing in the context of large-scale image classification.

4.6 Discussion

The main objective of this thesis is to implement a fresh classifier by breeding the existing metric learning algorithms, and hashing techniques in the context of large-scale image database classification. Several techniques of supervised classification of images depend on both the representation of local features of images, and metric distance to calculate the similarity (or distance) between images. Recently many studies have shown the interest to learn a metric rather than use simple metric. This learning approach is called *metric learning*. Because of this technique is relatively new, we are interested to use in the context of large-scale image classification.

The main aim of metric learning algorithm is to learn the distance measure using Mahanalobis metric distance by considering side information; and semi-definite

programming to use this information. There are some metric learning algorithms in the supervised classification context. Both theoretical and experimental test are used to choose one algorithm which performs better and improve the generic kNN classification significantly. Finally, we choose LMNN. As shown the experimental results in the previous sections, it improves the kNN classification accuracy rate significantly.

Due to the size of our input dataset, it is not feasible to use only metric learning algorithm to answer the objectives of this thesis. There are lots of indexing techniques instead of using exhaustive approach which is very expensive. Out of these indexing techniques, we use the simplest, intelligent and inexpensive indexing technique called locality sensitive hashing (LSH). Each metric distance and similarity measure has indexing functions. In this thesis, both Cosine similarity and *p*-stable distribution (Euclidean metric distance, E2LSH) hashing functions are used to build hash table.

As presented in the aforementioned lines, LMNN improves the kNN classification accuracy rate significantly. The major contribution of this thesis is to bring in LMNN improvement into both Cosine and *p*-stable distribution (Euclidean metric distance) hashing functions.

Finally, evaluate our solution using two methods: computational complexity and query accuracy. Computation complexity comprises the basic steps to build hash table and access ANN from the hash table. Similar to kNN classification accuracy, we use Query accuracy rate to evaluate the classification accuracy performance.

Based on the experimental results E2LSH outperforms both unlearned and learned Cosine similarity hashing. When we plug LMNN into E2LSH (LMNN +E2LSH), improves the original *p*-stable distribution hashing (Euclidean metric space) and even exhaustive methods using Euclidean distance measure. In the case of Cosine similarity, learned hashing functions improves the performance of randomized hashing function without metric learning. Generally, we concluded that LMNN metric algorithm has a competence to improve the performance of hashing in the context of large-scale image classification.

Chapter 5: Final Planning

Due to shortage of time and setup to achieve goals which are presented under Chapter one has changed. At the early stage, the plan was to deal supervised classification of images based on both: on the representation of local features of the image, and on the metric used to calculate the similarity (or distance) between the images. Following the above plan, to evaluate the performance of the solution and provide a distributed implementation. Because of the aforementioned reasons, we have decided to focus on the second technique of research direction and evaluate the performance, and skipped the distributed implementation. As a result, the initial planning has affected. Moreover, this research topic is new for the company. In other word, there was no any ongoing research under this topic. Therefore, it is a breakthrough and plays role to give direction for the coming related research topics in the company.

There are basic tasks which are not changed from initial plan rather than updating their time interval to deal with. These tasks are

- Literature survey's time interval never changed since it is the pillar of the research to make state-of-the-art. However, we have narrowed the area of survey towards studying metric learning algorithms, and machine learning algorithm's performance evaluation techniques. Following this task, due to large-scale of image database only learning distance does not enough to achieve the objective. Thus, studying dimension reduction and indexing techniques has got attention to achieve the research objectives.

- Studying metric learning algorithms and choose one algorithm using theoretical and experimental ground was taking place to achieve the objectives of this research

- Finally, the selected metric learning is used to learn the distance and plug into the indexing technique to maximize the accuracy of classification. To validate the proposed method, performance evaluation was taking place.

The thesis write up is another major step of the thesis to document all the phases and make sure to understand the flow of this research.

ID	Task Name	Start	Finish	Duration
1	Literature Review	2/3/2014	7/4/2014	110d
2	Study Metric learning techniques	2/13/2014	4/3/2014	36d
3	Compare and choose metric learning algorithm which perform better	3/31/2014	4/25/2014	20d
4	Study and choose dimension reduction algorithm	4/23/2014	5/6/2014	10d
5	Study and choose indexing techniques	5/5/2014	5/20/2014	12d
6	Implement the selected indexing technique	5/20/2014	6/13/2014	19d
7	Evaluation techniques implementation	6/11/2014	6/18/2014	6d
8	Experimental test	6/23/2014	7/4/2014	10d
9	Thesis Write up	2/10/2014	7/4/2014	105d
10	Correcting comments	7/7/2014	8/1/2014	20d

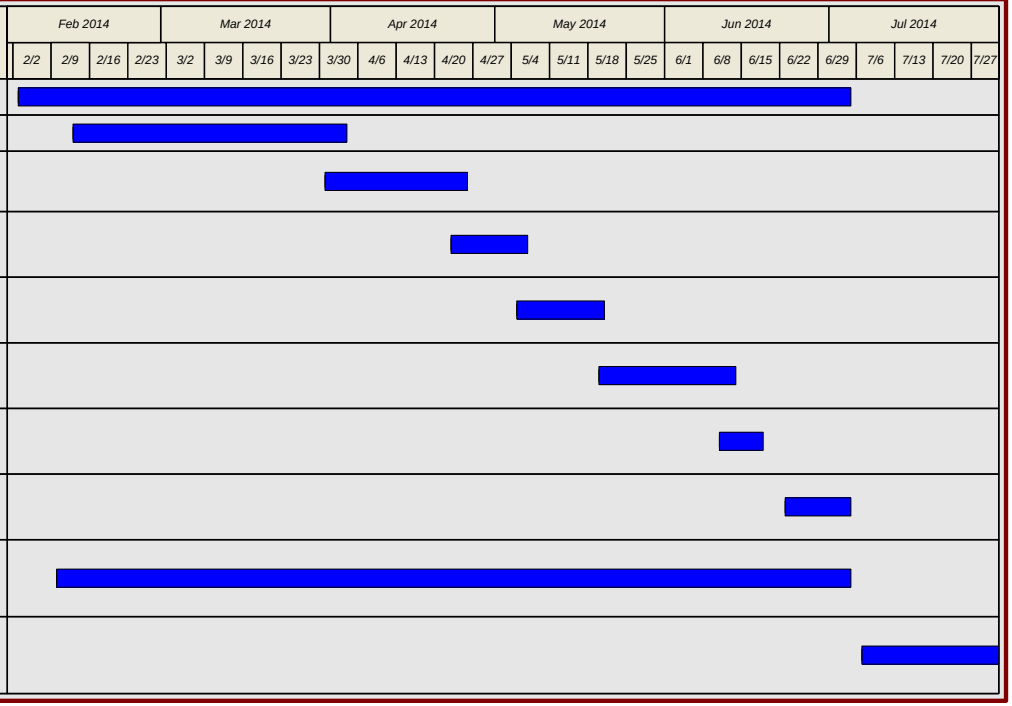


Figure 5-1: Final Planning

Chapter 6

6.1 Conclusion and Future Directions

In this thesis, we try to show the impact of metric learning over the generic kNN classification problem, and improve its performance significantly. Mahalanobis metric distance is the distance measure used in metric learning. Unlike Euclidean distance measure, Mahalanobis does use statistical distribution of the data. This distance measure problem is led by semi-definite programming. Due to its huge impact, in the past few years it has been getting huge attention in the area of large-scale image classification. Therefore, we are interested to dive into this area of research.

To gain the benefits of metric learning algorithm, we use it in this project to answer the objective of the thesis. To apply this algorithm, we focus on all of the metric learning algorithms which are driven by nearest neighbor approach since the objective of the thesis is to classify large-scale image database. To choose the algorithm which performs better than other, we use both theoretical and experimental evaluations. Based on these two evaluations, Large Margin Nearest Neighbor (LMNN) generally performs very well in practice. Due to the large dimension of our input dataset, its computation is very expensive and cause overfitting. To reduce computational time and avoid overfitting, we pre-process the input using Principle Component Analysis (PCA) to reduce the dimension of the data after normalizing each dimension of the input data using either *z-score* or *min-max* normalization technique.

Even though LMNN improves the generic kNN classification performance, it is infeasible to use it in a large-scale database exhaustively without indexing technique. There are various indexing solutions. Out of these, the most simple, intelligent and easy to implement is *Locality Sensitive Hashing* (LSH). This hashing technique is used under different similarity and distance measure metrics. In this project, we use both Cosine similarity and *p-stable* (Euclidean metric space) distribution hashing techniques. The main contribution of this project is to drag metric learning algorithm into hashing technique to gain the benefits. Finally, we evaluate the performance of our solution.

Generally, metric learning algorithms are used to learn the distance metric either to classify or cluster data points. Based on the experimental results, LMNN improves the kNN classification accuracy, and also when we use it in LSH; it improves the performance of generic hashing technique significantly. Therefore, we recommend metric learning algorithm if classification of accuracy is the main goal of the project.

6.2 Future directions

In addition to metric learning, the other major topic in image classification is feature extraction. Due to lack of time, in this thesis we focus only in the metric learning since it is new fashion for the past few years. To get full potential of metric learning, feature extraction plays great role by representing a given input data. We propose to extend this research work by adding feature extraction technique over this work.

To prepare input from images feature extraction algorithms are playing very important role. Therefore, it is vital to make our contribution complete.

On the other hand, we recommend further study to use other LMNN extension algorithms which are using different approach to resolve its drawbacks and compare results with this thesis.

Finally, we propose to advance this study using unsupervised metric learning algorithms. It is one of the hot topics in the current metric learning area of research.

Bibliography

[1] K. E. A. Van De Sande, T. Gevers, and C. G. M. Snoek, "Evaluating color descriptors for object and scene recognition," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 32, pp. 1582-1596.

[2] D. Ping Tian, "A Review on Image Feature Extraction and Representation Techniques," *International Journal of Multimedia & Ubiquitous Engineering*, vol. 8, pp. 385-395, 2013.

[3] A. l. Bellet, A. Habrard, and M. Sebban, "Good edit similarity learning by loss minimization," *Machine learning*, vol. 89, pp. 5-35, 2012.

[4] J. V. Davis, B. Kulis, P. Jain, S. Sra, and I. S. C. Dhillon, "Information-theoretic metric learning," in *Proceedings of the th international conference on Machine learning*, 2007, pp. 209-216.

[5] J. Goldberger, S. Roweis, G. Hinton, and R. Salakhutdinov, "Neighbourhood components analysis," 2014/02/06/23:00:59 2004.

[6] S. Shalev-Shwartz, Y. Singer, and A. Ng, "Online and batch learning of pseudo-metrics," in *Proceedings of the twenty-first international conference on Machine learning, ICML*, 2004.

[7] K. Weinberger, J. Blitzer, and L. Saul, "Distance metric learning for large margin nearest neighbor classification," *Advances in neural information processing systems*, vol. 18, 2014/02/21/07:59:28 2006.

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

[9] A. l. Bellet, A. Habrard, and M. Sebban, "A Survey on Metric Learning for Feature Vectors and Structured Data," *arXiv:1306.6709 [cs, stat]*, 2014/02/06/11:58:06 2013.

[10] K. Q. Weinberger and L. K. Saul, "Distance metric learning for large margin nearest neighbor classification," *The Journal of Machine Learning Research*, vol. 10, pp. 207-244, 2014/02/06/23:00:59 2009.

[11] H. Jegou, M. Douze, and C. Schmid, "Product quantization for nearest neighbor search," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 33, pp. 117-128, 2011.

[12] M. Muja and D. G. Lowe, "Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration," in *VISAPP (1)*, 2009, pp. 331-340.

[13] S. Meiser, "Point location in arrangements of hyperplanes," *Information and Computation*, vol. 106, pp. 286-303, 1993.

[14] K. L. Clarkson, "A randomized algorithm for closest-point queries," *SIAM Journal on Computing*, vol. 17, pp. 830-847, 1988.

[15] A. Andoni, P. Indyk, H. L. Nguyen, and I. Razenshteyn, "Beyond locality-sensitive hashing," *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, 2013.

[16] P. Indyk and R. Motwani, "Approximate nearest neighbors: towards removing the curse of dimensionality," in *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, 1998, pp. 604-613.

[17] W. Dong, Z. Wang, M. Charikar, and K. Li, "Efficiently matching sets of features with random histograms," in *Proceedings of the 16th ACM international conference on Multimedia*, 2008, pp. 179-188.

[18] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni, "Locality-sensitive hashing scheme based on p-stable distributions," in *Proceedings of the twentieth annual symposium on Computational geometry*, 2004, pp. 253-262.

[19] R. Pingdom, "Report: Social network demographics in 2012," *Tech Blog*, pp. http://royal.pingdom.com/2013/01/16/internet-2012-in-numbers/, 2012.

[20] C. C. Aggarwal and C. Zhai, "A survey of text classification algorithms," in *Mining text data*: Springer, 2012, pp. 163-222.

[21] L. M. Manevitz and M. Yousef, "One-class SVMs for document classification," *The Journal of Machine Learning Research*, vol. 2, pp. 139-154, 2002.

[22] D. Jayakody, "Automatic Video Classification," *Don Jayakody, San Jose State University, Computer Science*, p. http://scholarworks.sjsu.edu/etd_projects/45/, 2009.

[23] P. Kamavisdar, S. Saluja, and S. Agrawal, "A Survey on Image Classification Approaches and Techniques," *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 2, 2013.

[24] M. Slaney and M. Casey, "Locality-sensitive hashing for finding nearest neighbors [lecture notes]," *Signal Processing Magazine, IEEE*, vol. 25, pp. 128-131, 2008.

[25] B. Shaw, B. Huang, and T. Jebara, "Learning a distance metric from a network," in *Advances in Neural Information Processing Systems*, https://papers.nips.cc/paper/4392-learning-a-distance-metric-from-a-network.pdf, 2011, pp. 1899-1907.

[26] M. E. Taylor, B. Kulis, and F. Sha, "Metric learning for reinforcement learning agents," in *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, 2011, pp. 777-784.

[27] B. McFee, L. Barrington, and G. Lanckriet, "Learning content similarity for music recommendation," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 20, pp. 2207-2218, 2012.

[28] M. Norouzi, D. Fleet, and R. Salakhutdinov, "Hamming distance metric learning," in *Advances in Neural Information Processing Systems 25*, 2012, pp. 1070-1078.

[29] X. Ben, W. Meng, R. Yan, and K. Wang, "An improved biometrics technique based on metric learning approach," *Neurocomputing*, vol. 97, pp. 44-51, 2012.

[30] M. T. Law, C. S. Gutierrez, N. Thome, and S. Gancarski, "Structural and visual similarity learning for Web page archiving," in *Content-Based Multimedia Indexing (CBMI), 2012 10th International Workshop on*, pp. 1-6.

[31] R. m. Lajugie, S. Arlot, and F. Bach, "Large-Margin Metric Learning for Partitioning Problems," *http://arxiv.org/abs/1303.1280*, vol. 1, p. http://arxiv.org/pdf/1303.1280v1.pdf, 2013.

[32] P. Jain, B. Kulis, and K. Grauman, "Fast image search for learned metrics," in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, 2008, pp. 1-8.

[33] J. H. Friedman, J. L. Bentley, and R. A. Finkel, "An algorithm for finding best matches in logarithmic expected time," *ACM Transactions on Mathematical Software (TOMS)*, vol. 3, pp. 209-226, 1977.

[34] J. L. Bentley, "Multidimensional divide-and-conquer," *Communications of the ACM*, vol. 23, pp. 214-229, 1980.

[35] T. Liu, A. W. Moore, A. G. Gray, and K. Yang, "An Investigation of Practical Approximate Nearest Neighbor Algorithms," in *NIPS*, 2004, pp. 32-33.
[36] A. Gionis, P. Indyk, and R. Motwani, "Similarity search in high dimensions via hashing," in *VLDB*, Edinburgh, Scotland,, 1999, pp. 518-529.
[37] R. Weber, H.-J. r. Schek, and S. C. V. Blott, "A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces," 1998, pp. 194-205.
[38] K. Grauman and R. Fergus, "Learning binary hash codes for large-scale image search," in *Machine Learning for Computer Vision*: Springer, 2013, pp. 49-87.
[39] R. De Maesschalck, D. Jouan-Rimbaud, and D. s. L. Massart, "The mahalanobis distance," *Chemometrics and Intelligent Laboratory Systems,* vol. 50, pp. 1-18, 2000.
[40] B. Kulis and K. Grauman, "Kernelized locality-sensitive hashing," *Pattern Analysis and Machine Intelligence, IEEE Transactions on,* vol. 34, pp. 1092-1104, 2014/03/05/15:38:25.
[41] R. O. Duda, "Pattern Recognition for HCI," *Department of Electrical Engineering San Jose State University,* p. http://www.cs.princeton.edu/courses/archive/fall08/cos436/Duda/PR_Mahal/PR_Mahal.htm, 1997.
[42] S. P. Boyd and L. Vandenberghe, *Convex optimization*: Cambridge university press, 2004.
[43] "Machine learning," *Wikipedia,* p. http://en.wikipedia.org/wiki/Machine_learning, 2014.
[44] K. Crammer and Y. Singer, "On the algorithmic implementation of multiclass kernel-based vector machines," *The Journal of Machine Learning Research,* vol. 2, pp. 265-292, 2002.
[45] B. Schölkopf and A. J. Smola, *Learning with kernels: Support vector machines, regularization, optimization, and beyond*: MIT press, 2002.
[46] T. Cover and P. Hart, "Nearest neighbor pattern classification," *Information Theory, IEEE Transactions on,* vol. 13, pp. 21-27, 1967.
[47] J. Cao, M. Ahmadi, and M. Shridhar, "Recognition of handwritten numerals with multiple feature and multistage classifier," *Pattern Recognition,* vol. 28, pp. 153-160, 1995.
[48] D. Lu and Q. Weng, "A survey of image classification methods and techniques for improving classification performance," *International journal of Remote sensing,* vol. 28, pp. 823-870, 2007.
[49] J. A. Hartigan and M. A. Wong, "Algorithm AS 136: A k-means clustering algorithm," *Applied Statistics,* pp. 100-108, 1979.
[50] U. Von Luxburg, "A tutorial on spectral clustering," *Statistics and computing,* vol. 17, pp. 395-416, 2007.
[51] H. B. Barlow, "Unsupervised learning," *Neural computation,* vol. 1, pp. 295-311, 1989.
[52] R. D. Short and K. Fukunaga, "The optimal distance measure for nearest neighbor classification," *Information Theory, IEEE Transactions on,* vol. 27, pp. 622-627, 2014/02/12/10:55:02 1981.
[53] J. H. Friedman, "Flexible metric nearest neighbor classification," *Unpublished manuscript available by anonymous FTP from playfair. stanford. edu (see pub/friedman/README),* 2014/02/12/10:54:31 1994.
[54] T. Hastie and R. Tibshirani, "Discriminant adaptive nearest neighbor classification," *Pattern Analysis and Machine Intelligence, IEEE Transactions on,* vol. 18, pp. 607-616, 2014/02/12/10:55:35 1996.
[55] A. Globerson and S. C. N. Roweis, "Metric learning by collapsing classes," in *Advances in neural information processing systems*, 2005, pp. 451-458.
[56] J. Blitzer, K. Q. Weinberger, and L. K. C. Saul, "Distance metric learning for large margin nearest neighbor classification," in *Advances in neural information processing systems*, 2005, pp. 1473-1480.
[57] S. Parameswaran and K. Q. Weinberger, "Large Margin Multi-Task Metric Learning," in *NIPS*, 2010, pp. 1867-1875.

[58] K. Q. Weinberger and L. K. Saul, "Fast solvers and efficient implementations for distance metric learning," in *Proceedings of the 25th international conference on Machine learning*, 2008, pp. 1160-1167.
[59] D. Kedem, Z. E. Xu, and K. Q. Weinberger, "Gradient Boosted Large Margin Nearest Neighbors," *NIPS*, vol. 25, pp. 2582-2590, 2012.
[60] D. Kedem, S. Tyree, K. Q. Weinberger, F. Sha, and G. R. G. Lanckriet, "Non-linear Metric Learning," in *In Advances in Neural Information Processing Systems (NIPS)*, 2012, pp. 2582-2590.
[61] L. Torresani and K.-c. Lee, "Large margin component analysis," *Advances in neural information processing systems*, vol. 19, 2014/03/13/16:28:20 2007.
[62] C. J. C. Burges, "Dimension Reduction: A Guided Tour," *Found. and Trends in Machine Learning*, vol. 2, pp. 275-365, 2010.
[63] H. Abdi and L. J. Williams, "Principal component analysis," *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 2, pp. 433-459, 2010.
[64] M. S. Charikar, "Similarity estimation techniques from rounding algorithms," in *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, 2002, pp. 380-388.
[65] M. X. Goemans and D. P. Williamson, "Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming," *Journal of the ACM (JACM)*, vol. 42, pp. 1115-1145, 1995.
[66] A. Andoni and P. Indyk, "E 2 LSH 0.1 User Manual," 2005.
[67] V. M. Zolotarev, *One-dimensional stable distributions* vol. 65: American Mathematical Soc., 1986.
[68] W. Zhang, K. Gao, Y.-d. Zhang, and J.-t. Li, "Data-oriented locality sensitive hashing," in *Proceedings of the international conference on Multimedia*, 2010, pp. 1131-1134.
[69] J. Ullman, "Data Mining," p. http://infolab.stanford.edu/~ullman/mining/2009/, 2009.
[70] J. Han, M. Kamber, and J. Pei, *Data mining: concepts and techniques*, 3 ed.: Morgan kaufmann, 2006.

EURA NOVA

EURA NOVA[10] is a young company founded in 2008, established and based in Belgium. The two founding partners are Eric Delacroix and Hervé Bath. Now, more than 30 leading consultants and reputed researchers are working. The area of work focuses:

1. Provision of state-of-the-art consulting services to clients
2. Management of an independent research centre
3. Development of a product innovation

[10] http://euranova.eu/ (browse this link for more information)