Exploring geometrical structures in high-dimensional computer vision data

Xiping Fu

a thesis submitted for the degree of

Doctor of Philosophy

at the University of Otago, Dunedin,

New Zealand.

27 January 2016

Abstract

In computer vision, objects such as local features, images and video sequences are often represented as high dimensional data points, although it is commonly believed that there are low dimensional geometrical structures that underline the data set. The low dimensional geometric information enables us to have a better understanding of the high dimensional data sets and is useful in solving computer vision problems.

In this thesis, the geometrical structures are investigated from different perspectives according to different computer vision applications. For spectral clustering, the distribution of data points in the local region is summarised by a covariance matrix which is viewed as the Mahalanobis distance. For the action recognition problem, we extract subspace information for each action class. The query video sequence is labeled by information regarding its distance to the subspaces of the corresponding video classes. Three new algorithms are introduced for hashing-based approaches for approximate nearest neighbour (ANN) search problems, *NOKMeans* relaxes the orthogonal condition of the encoding functions in previous quantisation error based methods by representing data points in a new feature space; *Auto-JacoBin* uses a robust auto-encoder model to preserve the geometric information from the original space into the binary codes; and *AGreedy* assigns a score, which reflects the ability to preserve the order information in the local regions, for any set of encoding functions and an alternating greedy method is used to find a local optimal solution.

The geometric information has the potential to bring better solutions for computer vision problems. As shown in our experiments, the benefits include increasing clustering accuracy, reducing the computation for recognising actions in videos and increasing retrieval performance for ANN problems.

Acknowledgements

My PhD journey has been helped by many great people who have been part of this invaluable experience. These people have endeared me with pivotal life-long lessons of which I am extremely grateful.

First of all, I would like to express my sincere gratitude to my supervisors Brendan McCane, Steven Mills and Michael Albert. I must acknowledge that they truly are the best supervision team. They provided professional, comprehensive and caring supervision which is important for me to conduct my research. Our weekly meetings were full of insightful discussions and I was inspired and encouraged to explore the high dimensional world. I am also grateful to Shawn Martin and Lech Szymanski who gave me lots of useful advice and contributed to my PhD research.

I would like to thank all members in Graphics and Vision research group including Nabeel Khan, Umair Khan, Hamza Bennani, Maria Mikhisor, Jordan Campbell, Rassoul Mesbah, and Tapabrata Chakraborti. The lab life was friendly and enjoyable, and we had vital discussions which undoubtedly contributed to this research.

Last but not least, I would like to thank my parents and family members, and especially my wife, Xiaorong Wang, for her support, patience and encouragements during the past three years. I really appreciate Xiaorong's willingness to care for our parents as well as our lovely young daughter.

Contents

List of Tables

List of Figures

Notations

Most of the notations are defined when they are introduced for the first time and clear in the context. For quick reference, here are the common symbols used throughout the thesis:

N: the number of data points in a data set

x_i: the i-th data point in a data set

X: the data set matrix where each column is a data point

D: the dimension of the high dimensional space

d: the dimension of the low dimensional space

$\mathcal{R}^d$: the d dimensional Euclidean space

B: the binary code matrix where each column is a binary code of a data point

$\mathbf{1}$: the all one column vector

I: the identity matrix

sign: the sign function which returns -1 or 1 as entries depending on the sign of the input entries

$\mathcal{M}$: the general manifold space

$\exp$: the exponential map in Riemannian manifold

$\log$: the logarithmic map in Riemannian manifold

$\mathcal{G}(d, D)$: the Grassmannian manifold which is a space of all the d dimensional subspaces of the D dimensional Euclidean space

$\emptyset$: the empty set

$T_p(\mathcal{M})$: the tangent space of manifold $\mathcal{M}$ at p

$||A||_F$: the Frobenius norm of matrix A

Chapter 1

Introduction

In computer vision, we are often confronted with high dimensional data sets. For example, the local image descriptor SIFT (Lowe, 2004) is represented as a 128D feature vector and the global image descriptor GIST (Oliva and Torralba, 2001) can be represented as a 960D feature vector. The high dimension of the data sets poses challenges for us to solve the corresponding computer vision problems including clustering, classification, and visualisation tasks. The performance of machine learning tasks deteriorates quickly as the dimension of the data set increases, and this phenomenon is often referred as 'the curse of dimensionality' (Donoho, 2000).

Although the data sets are represented in high dimensional space, it is believed that they have some low dimensional intrinsic structures due to the intrinsic freedom of the computer vision objects. As a case in point, suppose we take $1,000$ pictures of a cup. The cup is on a table, and all of the pictures are taken under same camera parameters as well as same environment such as illumination conditions. The only difference is the camera positions which are sampled from a circle. On one hand, only one parameter (radian) is enough to represent the relationship between all of the pictures, i.e., each picture can be determined by a scalar. On the other hand, we might represent each image by a high dimensional descriptor such as a 960D GIST vector. Assuming that the high dimensional representation is faithful, we expect some low dimensional structure inside the data points in the high dimensional Euclidean space. In addition, for the data set that consists of high dimensional vectors of the $1,000$ pictures, the 1D geometric information enables us to have a better understanding of the distribution of the data points in the high dimensional space. Such information can be further used for solving other computer vision tasks such as visualising the data set in low dimensional space.

From the discussion above, we can see that geometric information from the data set plays a pivotal role in solving different computer vision problems. Manifold learning has many successful techniques for discovering the intrinsic geometric information of the data set, including *Isomap* (Tenenbaum, De Silva, and Langford, 2000) and *LLE* (Roweis and Saul, 2000) to learn the low dimensional representation of the data set. For instance, the main task of *Isomap* is to learn the low dimensional representation of the original data set. The data points in the low dimensional space are assumed to preserve the geodesic distance between data points in the original space, i.e., the low dimensional representation is faithful to the geometric information. Thus, the distance information in the low dimensional space has a better semantic meaning than the distance calculated in the original space directly. In addition, the geometric information can also be used as additional information for solving computer vision problems. For instance, the curvature is used to describe the distribution of the data points in a local region, and this information has been used for clustering the data set (Kim, Tompkin, and Theobalt, 2013).

1.1 Goal and objectives

The quality of the approximation has an intimate relationship with the information we obtained from the corresponding problems. As discussed in Jordan and Mitchell (2015), the key feature of different machine learning algorithms is that they are approximating some functions in essence. This means that the more information we obtain from the data set, the better mathematical model we might build for fitting real-world computer vision problems.

In this thesis, our main goal is to find structures, i.e., explore appropriate geometric information, in high dimensional data sets for computer vision problems. Although it is commonly assumed that data sets have low dimensional structures in high dimensional spaces, the low dimensional structures are unclear for real world data sets. This is partly due to the variety of the computer vision objects and the noise which is introduced when we describe the objects. The structure can be any kind of information which helps us to have a better understanding of the data set from the geometric perspective. For example, the covariance of a data set in the local region helps us to figure out how the data points are distributed. Thus, the covariance information is also a kind of structure or geometric information of the data set, and it can be used for solving real world problems.

According to different application scenarios we focus on, our objectives consist of three computer vision problems: spectral clustering for intersecting multiple manifolds problems, video classification based on representation in the product of Grassmannian manifolds, and hashing-based approximate nearest neighbour (ANN) search problems.

- **Spectral clustering for intersecting multiple manifolds problems.**

  Clustering is widely used in fields such as computer vision, information science and machine learning. The goal of clustering is to divide the data set into different clusters based on some similarity information between the data points (Rodriguez and Laio, 2014). There are many approaches available (Berkhin, 2006). As an example, K-means is a simple algorithm that partitions a data set into K groups by minimising intra-cluster Euclidean distance.

  In the past decades, spectral clustering algorithms have been developed to overcome some of the shortcomings of traditional clustering methods (Shi and Malik, 2000; Ng, Jordan, and Weiss, 2002; Von Luxburg, 2007). Traditional algorithms often assume normally distributed clusters, while spectral algorithms are capable of clustering non-Gaussian, non-linear data. Spectral clustering works by representing data similarity as a weighted graph. Each point in the data set is a node in the graph, and the non-zero edges in the graph connect similar data points. This graph can be mapped into a new space by computing eigenvectors of an associated matrix. Finally, K-means is used to cluster the data in the new space.

- **Video classification based on representation in the product of Grassmannian manifolds.**

  Human action recognition is a challenging research topic in computer vision with numerous applications. For instance, it can be used to construct gesture based human computer interfaces or build an automated surveillance system to detect abnormal activities in public places (Aggarwal and Ryoo, 2011).

  The representation of video sequences plays a pivotal role for practical action recognition systems. The raw pixels of the video sequence are the starting point of different representations. For example, both the optical flow and the gradient information can be extracted based on the set of pixel values (Chaudhry, Ravichandran, Hager, and Vidal, 2009; Lin, Jiang, and Davis, 2009; Shechtman and Irani, 2005; Dalal, Triggs, and Schmid, 2006; Niebles, Wang, and Fei-Fei, 2008). The raw pixel can also be used to represent video sequences directly. The difference between consecutive video sequences has an intimate relationship with linear space. Thus, Grassmannian manifolds are often used to summarise the video sequences (Turaga, Veeraraghavan, and Chellappa, 2008; Lui, Beveridge, and Kirby, 2010). After representation, the distance information in the manifold space or other geometric information of the data points is used for the action recognition task.

- **Hashing-based approaches for ANN search problems.**

  Recent years have witnessed a surge of research interest in hashing methods which return ANN data points. It has been shown that ANN often has good enough performance for many real world applications including feature matching (Brown and Lowe, 2003), image retrieval (Frome, Singer, Sha, and Malik, 2007) and object recognition (Torralba, Fergus, and Weiss, 2008). For example, in content-based image retrieval (CBIR), the task is to retrieve similar images when given a query image. This is often done by representing all of the images as points in a specific space, and then retrieving the nearest neighbour points as similar images. Naive exhaustive searching is linear in the number of images in the collection and becomes infeasible for very large collections. Even specialised data structures such as KD-trees deteriorate to linear search complexity or worse if the dimensionality of the data is large (Weber, Schek, and Blott, 1998). Since computer vision problems often have very large collections and high dimensional data, approximation algorithms are of interest.

  The main idea of hashing is to encode data points into binary codes which enables fast retrieval and reduces storage space. As a case in point, one 960D GIST feature takes $960 \times 4 = 3840$ bytes if it is stored as a floating point vector, while it only occupies $128/8 = 16$ bytes if it is represented as a point in 128D Hamming space. Further, calculating the distance between two points in Hamming space is very quick since it only involves a bitwise XOR operation followed by a bit count.

1.2 Research methods

Although the tasks have a variety of formats among different computer vision problems, the final mathematical models are to deal with high dimensional data sets in essence. The main difficulty is to decide what kind of geometric information can be used. For specific computer vision problems, we have to analyse the problem and get a better understanding of where the bottleneck is.

We explore the geometric information from two perspectives. The first one is to investigate whether techniques from related fields such as the mathematical concepts from the theory of manifolds and techniques from manifold learning can be used. Manifolds are a well-studied mathematical model and provide nonlinear generalisations of many concepts in Euclidean space. In the past decades, we can see lots of successful works have ideas rooted in manifold concepts. One of the main purpose of manifold learning is to uncover the manifold structure in the data set, and then use the information from the manifold structure to solve real-world problems. Thus, the concepts and techniques from these two fields are our first priority to investigate.

The second perspective is to explore appropriate geometric information for corresponding computer vision problems. In addition to the techniques provided from the field of manifolds and manifold learning, we should also pay attention to new geometric information for specific computer vision problems. Here the new geometric information is defined in a broader sense including any kind of information related to the distribution information of a data set. The new geometric information is tailored according to the need for addressing the corresponding problems. Thus it has the potential to bring better solutions. Considering a set of data points, the order information of the data points according their distances to some fixed point can be viewed as a kind of geometric information extracted from the data. In later sections, we will see that the order information can be triangulated to learn encoding functions for hashing methods.

1.3 Contributions

For spectral clustering, we propose to use the adaptive Mahalanobis distance for selecting the neighbourhood data points with the aim to build a better KNN graph. For video sequence classification, we propose to summarise the information of each action class in the manifold space. For the hashing-based approach, we have proposed three new hashing methods, *NOKMeans*, *Auto-JacoBin* and *AGreedy*, from different perspectives. The first two hashing methods aim to learn the encoding functions directly, and the third one selects a set of encoding functions from a pool generated by any of the previous hashing methods. The corresponding background and the contributions are as follows:

- **Adaptive Mahalanobis distance for spectral clustering (Fu, Martin, Mills, and McCane, 2013).**

  Spectral clustering tends to fail when the underlying manifolds are very close to each other and/or they intersect. We propose an improvement to spectral clustering algorithms using adaptive neighbourhoods computed using Mahalanobis distance. The distribution information of data points in local region is summarised by a covariance matrix which is viewed as a Mahalanobis distance and is used to relocate data points in the local region. By repeating the learning in the local region, we aim to improve the quality of the similarity between data points.

- **Summarising the geometric information of video sequences (Fu, McCane, Albert, and Mills, 2013).**

  After representing video sequences in the product of Grassmannian manifolds space, we extract a subspace structure for each action class in the training data set. The subspace structure summarises the distribution information of the corresponding action class in the manifold space.

- **Labelling video sequence by its distance to the learned geometric information (Fu *et al.*, 2013).**

  With the information extracted from training data points in the manifold space, the query video sequence is labeled by the information of its distances to the subspaces of the corresponding video classes. The results on benchmark data sets show that the new approach takes less computing time compared to previous work in this area, and has similar or even higher recognition accuracy.

- **Relaxing the orthogonal condition in quantisation error based hashing approaches (Fu, McCane, Mills, and Albert, 2014).**

  The quantisation error based hashing approaches aim to reduce the quantisation error between binary vectors and data points in Euclidean space. One condition which is implicitly assumed is that the separating hyperplanes are mutually orthogonal. In order to increase the representation capability of the hyperplanes when used for indexing, we relax the orthogonality assumption without forsaking the alternate view of using cluster centres to represent the indexing partitions. This is achieved by viewing the data points in a space determined by their distances to the hyperplanes.

- **First order approximation of an ideal noise removing function**

  In order to preserve the geometric information of the data set, we consider an auto-encoder model which has the noise removing effect. The function (forward propagation) is defined with the property that it projects the data points near the manifold into the manifold wisely. We approximate this function by its first order approximation which has an intimate relationship with the data points in the local region.

- **Auto-encoder Jacobian binary hashing (*Auto-JacoBin*)**

  For learning the encoding functions, the optimisation objective consists of two components. The first component aims to preserve the geometric information of the data set. This is done by minimising the gap between the learned auto-encoder model and the learned first order approximation of an ideal noise removing function. The second is a constraint on the hidden features and it encourages the hidden features evenly distributed around the vertices of a hyper-cube in order to make full use of binary codes.

- **Scoring the encoding functions based on training data set (Fu, McCane, Mills, and Albert, 2015).**

  Most previous hashing methods are based on some mathematical assumptions and learn the encoding functions by solving the corresponding optimisation problems. Thus, selecting a set of encoding functions from a pool which is generated by a set of hashing methods might lead to a better quality of the encoding. Given a pool of bits, we propose to select a set of bits according to a quality measurement directly related to the large scale approximate nearest neighbour search problem. The higher score means the better quality of the encoding functions. An alternating greedy optimisation method is proposed to find a locally optimal solution.

1.4 Thesis overview

This thesis consists of eight chapters and one appendix which is used to provide the support material for Auto-JacoBin. The rest of the thesis is structured as follows.

Chapter 2 presents a brief introduction of manifolds and manifold learning. The techniques and concepts provide background for the work in later chapters.

The following five chapters are divided into three parts according to different computer vision problems. The first part is about spectral clustering for intersecting multiple manifolds problems. The main challenge for this kind of clustering task is caused by the fact that the data points from different classes intersect with or lie close to one another. We propose to find the neighbourhood data points iteratively by Mahalanobis distance. The second part is about video classification. Each video sequence is represented as a data point in the product of Grassmannian manifolds. In previous work, the classification is done by the nearest neighbour classifier which takes heavy computation when the size of the training data set is large. We propose to extract geometric information for each video class, and use this information for labelling the query video sequences.

The third part is about hashing-based ANN problems. Chapter 5 summarises our work in *NOKMeans*. The orthogonal assumption in quantisation error based hashing approaches is relaxed which enhances the partition capability of the encoding functions. Chapter 6 presents our work on *Auto-JacoBin*. Our main motivation is to investigate whether an auto-encoder model can be used for learning the encoding functions since the auto-encoder model is widely used for preserving information in data sets. Chapter 7 concerns an algorithm called *AGreedy*. We propose to evaluate the set of encoding functions, i.e., assign a score for any set of the encoding functions. In this way, the bit selection problem becomes an optimisation problem. According to the property of the bit selection problem, the alternating greedy optimisation method is used to find a locally optimal solution.

Finally, Chapter 8 contains the discussion of the work in this thesis as well as the directions for possible future research. Appendix A provides the detail of the gradients calculation for the algorithm in Chapter 6.

In Fig. 1.1, we summarise the different parts of the thesis into a graph structure in order to have a visualisation of the thesis.

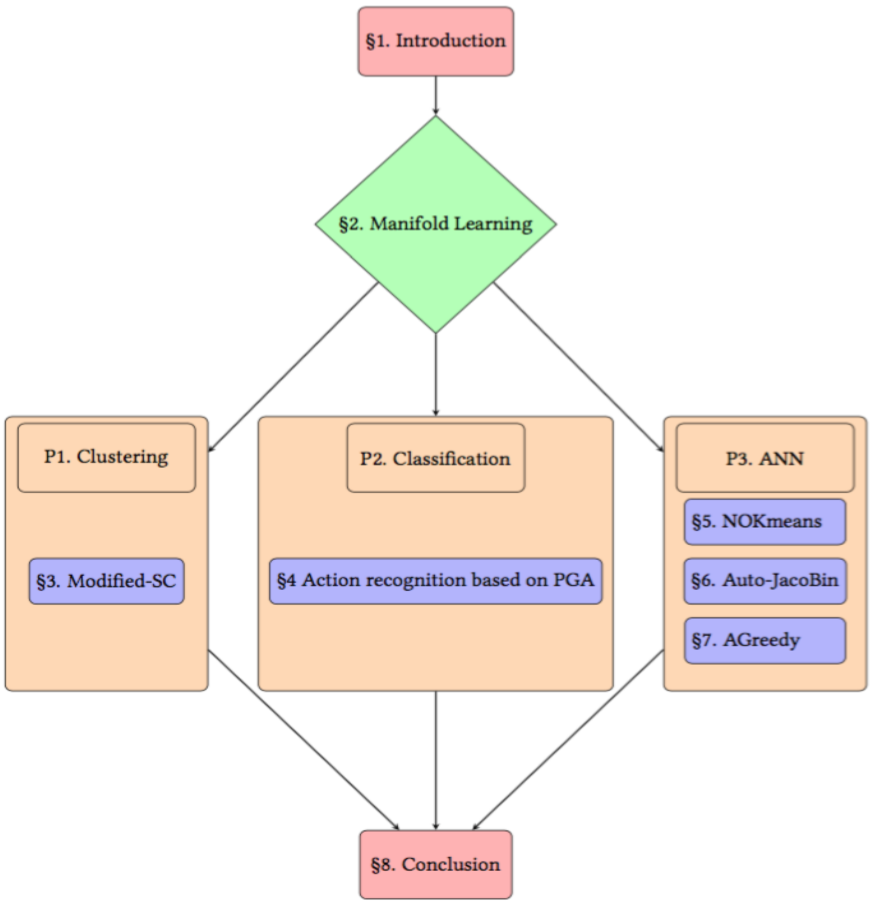


Fig. 1.1. Layout of the thesis and the relationship between different parts.

Chapter 2

Manifold learning background

2.1 Introduction

This chapter summarises the mathematical background of manifolds and their applications in manifold learning. Manifolds provide mathematical models and tools for us to explore geometric information of high dimensional data sets. For real-world high dimensional data sets, it is often believed that the data points are distributed on some low dimensional manifold. In practice, it is not easy to explore the explicit manifold structures inside the data set since the data points inevitably have noise, the data points are not well sampled or the number of data points is not enough to learn the manifold structure. Thus, the main task of manifold learning is to explore the implicit manifold structure of the data set. This kind of information helps us to understand high dimensional data sets, and can be used for solving different computer vision or machine learning problems.

In the following sections, we present some relevant concepts related to manifolds in order to have an intuition of the geometrical structures, and then we summarise some techniques, which illustrate how the geometric information is used in practice, in manifold learning.

2.2 Manifolds

In this section, some relevant concepts related to manifolds are summarised. The intuition behind these concepts is important to understand the manifold when it is embedded in some high dimensional space. We take the Grassmannian manifold as an example, and its related computations are presented to facilitate the illustrations in later chapters. In the next section, we will see that many manifold learning algorithms are motivated by the concepts from smooth manifolds.

2.2.1 Manifold concepts

In this subsection, the basic concepts of manifolds will be discussed in an informal fashion. This is so that the underlying intuition required to understand manifold learning can be developed. For the complete development of this branch of mathematics see some monographs (Lee, 2010, 2012; Absil, Mahony, and Sepulchre, 2009).

Denote $\mathcal{M}$ as a set of points in $\mathbb{R}^D$. Without any structure given to the set, it behaves as a container, i.e., it is a collection of data points and the different points are independent from each other. Take a set with point list $\mathcal{M} = \{p_1, p_2, p_3, \cdots\}$ as an example, the only information we can get is the membership of the set, i.e., for a given point x, we have the membership as either $x \in \mathcal{M}$ or $x \notin \mathcal{M}$.

The topology space $\mathcal{T}$ of $\mathcal{M}$ provides a way to enrich the structure of the point set. It is a collection of subsets of $\mathcal{M}$ and satisfies three conditions (Lee, 2010):

1. Both the empty set $\emptyset$ and $\mathcal{M}$ are in $\mathcal{T}$;
2. For any $s_1 \in \mathcal{T}$, $s_2 \in \mathcal{T}$, then $s_1 \cap s_2 \in \mathcal{T}$;
3. For any index set I, such that $\{s_i\}_{i\in I} \subset \mathcal{T}$, then $\cup_{i\in I} s_i \in \mathcal{T}$.

The topological space for the set helps to organise the ‘closeness’ relationship between members in the set. For example, if $\mathcal{T}$ consists of all of the subsets of $\mathcal{M}$, then for any two points $x_1, x_2 \in M$, there exist $U_1 = \{x_1\} \in \mathcal{T}$, $U_2 = \{x_2\} \in \mathcal{T}$, such that $x_1 \in U_1, x_2 \in U_2$, and $U_1 \cap U_2 = \emptyset$. This means any two distinct points can be separated by different elements from $\mathcal{T}$, whereas if $\mathcal{T}$ consists of only two subsets $\{\emptyset, \mathcal{M}\}$, there is no such separation guarantee for distinct points.

Each element in $\mathcal{T}$ is called an open subset of $\mathcal{M}$. In the following discussions, the topological space of Euclidean space $\mathbb{R}^D$ and its subsets are defined consistent with our common sense. The topological space $\mathcal{T}$ of $\mathbb{R}^D$ consists of all of the set A with the property: for any $x \in A$, there exists $\epsilon > 0$, such that the ball $\mathcal{B}(x, \epsilon) \subset A$. For any subset of $\mathbb{R}^D$, its topological space is induced by restricting $\mathcal{T}$ to the corresponding subset.

With topological space defined on U and V, a function f between them can be investigated. The function $f : U \to V$ is said to be continuous if for any open set s_v in V, $f^{-1}(s_v)$ is an open set in U. The f is called to be homeomorphism if it is bijective and both f and f^{-1} are continuous. When U and V are open subsets of Euclidean spaces, the f is said to be smooth (C^∞) if it is continuous and differentiable for any order, and the f is said to be a diffeomorphism if both f and f^{-1} are smooth.

With the concepts for functions, the structure of $\mathcal{M}$ is explored by functions defined on its subsets. A chart (U, f) of $\mathcal{M}$ is a homeomorphism $f : U \to V$ where $U \subset \mathcal{M}$ and V is an open subset of $\mathbb{R}^D$. An smooth atlas of $\mathcal{M}$ is a collection of charts $\mathcal{A} = \{(u_\alpha, \phi_\alpha)\}_{\alpha\in I}$, such that $\cup_{\alpha\in I} u_\alpha = \mathcal{M}$ and the different charts are compatible with each other, i.e., for any two (u_α, ϕ_α) and (u_β, ϕ_β), $\phi_\alpha \circ \phi_\beta^{-1}$ and $\phi_\beta \circ \phi_\alpha^{-1}$ are smooth.

With the definition of chart and atlas, local regions can be explored by some specific chart and the atlas enables that all of the regions can be explored. A smooth D-manifold $\mathcal{M}$ is a set with a maximal smooth atlas $\mathcal{A} = \{u_\alpha, \phi_\alpha\}_{\alpha\in I}$, i.e., for any smooth atlas $\mathcal{B}$, if $\mathcal{A} \subset \mathcal{B}$, then $\mathcal{A} = \mathcal{B}$, and for $\forall(u_\alpha, \phi_\alpha) \in \mathcal{A}$, ϕ_α maps u_α into a D dimensional open subset in $\mathbb{R}^D$. With the smooth atlas structure $\mathcal{A}$ on the manifold $\mathcal{M}$, the function f, which maps $\mathcal{M}$ into a Euclidean space, is called to be smooth if for any $p \in \mathcal{M}$, there exists an chart (U, ϕ) such that $p \in U$ and $f \circ \phi^{-1}$ is a smooth function (Lee, 2012). For functions, which map between two manifolds, the smoothness is defined similarly by the smooth atlas structures.

The definition of a smooth manifold helps us to understand the data set from local perspectives. A manifold is a set of points whose local structure is like that of $\mathbb{R}^D$. As a case in point, the unit sphere in $\mathbb{R}^3$ is a smooth 2-manifold. From the definition of smooth 2-manifold, we can see that the local patch from the sphere has similar properties as $\mathbb{R}^2$.

In multivariable calculus, the gradients and tangent plane can be defined analytically and can be visualised in Euclidean space. For manifolds, tangent vector and tangent space have their corresponding generalisations. The tangent vector can be defined implicitly by utilising functions on the manifold (Absil *et al.*, 2009). Denote $C_p^\infty(\mathcal{M})$ as the set of all the smooth functions which map the neighborhood of p into $\mathbb{R}$, and $f : \mathbb{R} \to \mathcal{M}$ is a smooth map. For all $g \in C_p^\infty(\mathcal{M})$, $g \circ f$ is a smooth map between some open subsets of $\mathbb{R}$, thus we have the corresponding gradient. A tangent vector ϵ for manifold $\mathcal{M}$ at p is defined as a map from $C_p^\infty(\mathcal{M})$ to $\mathbb{R}$, where the map is defined through an implicit curve $f : \mathbb{R} \to \mathcal{M}$, and

$$\epsilon(g) = \lim_{\tau\to 0} \frac{g \circ f(t+\tau) - g \circ f(t)}{\tau}. \tag{2.1}$$

The tangent space of $\mathcal{M}$ at p is the set of all tangent vectors at p, and it is a vector space with the same dimension as the manifold. Whereas the definition of tangent vector is implicit and abstract since it is a map on $C_p^\infty(\mathcal{M})$, the tangent space enables further exploration of the manifold. For example, with the inner product structure defined appropriately on tangent spaces $T_p(\mathcal{M})$, lots of concepts in Euclidean space can be generalised to manifold space.

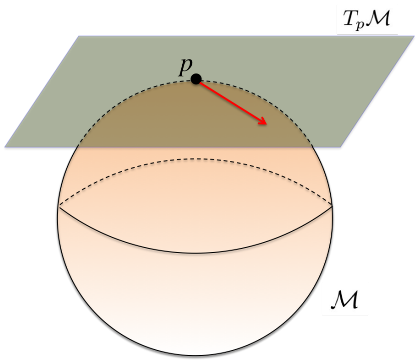


Fig. 2.1. Visualization of the tangent space $T_p(\mathcal{M})$. The tangent space consists of all of the tangent vectors at p. This figure is redrawn from Absil *et al.* (2009).

A Riemannian metric g defines an inner product on all of the tangent spaces and it is a smooth function when it is viewed as a function defined on $\{(\epsilon_p^1, \epsilon_p^2) \mid (\epsilon_p^1, \epsilon_p^2) \in T_p(\mathcal{M}) \times T_p(\mathcal{M}),\ p \in \mathcal{M}\}$ and $g(\epsilon_p^1, \epsilon_p^2) = \langle \epsilon_p^1, \epsilon_p^2 \rangle_{g_p}$ (Absil *et al.*, 2009). The manifold $\mathcal{M}$ equipped with a Riemannian metric g is called a Riemannian manifold $(\mathcal{M}, g)$. With the inner product g_p on tangent space $T_p(\mathcal{M})$, the norm of the tangent vector ϵ_p is induced by the inner product:

$$||\epsilon_p||_g = (\langle \epsilon_p, \epsilon_p \rangle_{g_p})^{\frac{1}{2}}. \tag{2.2}$$

Since the tangent vector is a generalisation of the gradient in multivariable calculus, the length of a smooth curve is calculated by the length of the tangent vectors along the curve. Suppose $f : [a, b] \to \mathcal{M}$ is a piecewise smooth curve on a Riemannian manifold $(\mathcal{M}, g)$, the length of the curve is defined as

$$L(f) = \int_a^b ||\dot{f}(t)||_g dt, \tag{2.3}$$

where $\dot{f}(t)$ is the corresponding tangent vector induced by f at t.

With the definition of the length of a curve in the manifold, geodesic distance between two data points $p_1$and p_2 in the manifold is defined as the ‘shortest’ distance among all of the lengths of the curves which start from p_1 and end at p_2 (Absil *et al.*, 2009), i.e.,

$$D_g(p_1, p_2) = \inf_{f \in F} L(f), \tag{2.4}$$

where $F = \{f \mid f : [a, b] \to \mathcal{M}$ is a piecewise smooth curve, and $f(a) = p_1, f(b) = p_2\}$.

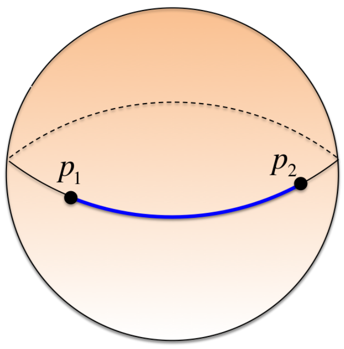


Fig. 2.2. Visualisation of the geodesic distance. The geodesic distance between two data points p_1 and p_2 is the minimum length of the curves between them.

The tangent space has an intimate relationship with the manifold in the local region. The exponential map, $\exp$, and the logarithmic map, $\log$, make connections between data points on the manifold and data points in the tangent space. Considering the tangent space $T_p(\mathcal{M})$ of the Riemannian manifold $(\mathcal{M}, g)$, for any $x \in T_p(\mathcal{M})$, there exists a unique (geodesic) curve such that $f(0) = p$, $\dot{f}(0) = x$ and $\ddot{f}(t) = 0$ (Absil *et al.*, 2009). Denote $p_2 = f(1)$, the exponential map $\exp$ and the logarithmic $\log$ are defined as

$$\exp(x) = p_2, \tag{2.5}$$

and

$$\log(p_2) = x. \tag{2.6}$$

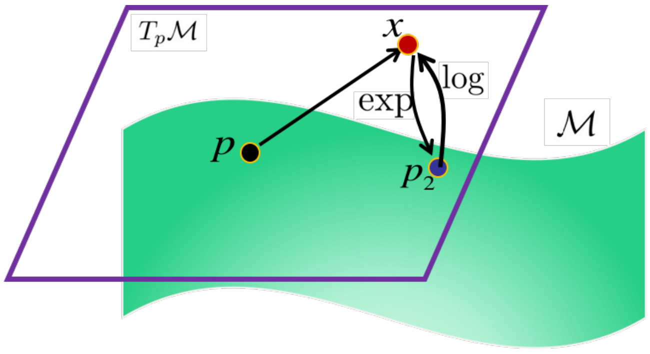


Fig. 2.3. Visualisations of the exponential map $\exp$ and the logarithmic map $\log$. The exponential map $\exp$ maps each tangent vector to the point in the nonlinear manifold. Both of these two points have the same distance to the origin of the tangent space. The logarithmic map $\log$, which is defined around the origin, is the inverse function of the exponential map. This figure is redrawn from Lee (2012).

Above, a number of concepts related to manifolds have been summarised. From a set of data points, with structures equipped gradually, the set becomes a manifold or Riemannian manifold, which enables us to explore the geometrical properties of the data set including the distance computation between two manifold data points and the local approximation of the manifold space. The main purpose of this section is to present the key concepts and intuitions behind manifolds.

2.2.2 Example: Grassmannian manifold

Grassmannian manifold has wide applications in computer vision. It has been used to represent video sequences (Turaga, Veeraraghavan, Srivastava, and Chellappa, 2011), a set of images with different poses or different illuminations (Harandi, Sanderson, Shirazi, and Lovell, 2011). In this section, take the Grassmannian manifold $\mathrm{Gr}(d, D)$ as an example, the key concepts, such as the tangent space, geodesic curve, exponential map $\exp$ and the logarithmic map $\log$, are summarised for this specific manifold space.

The Grassmannian manifold $\mathrm{Gr}(d, D)$ consists of all the d-dimensional subspaces in $\mathbb{R}^D$. Denote X' as the transpose of a matrix X. Since any matrix $X \in \mathbb{R}^{D\times d}$ with the property $X'X = I_d$ determines a unique d-dimensional subspace, $\mathrm{Gr}(d, D)$ is defined as

$$\mathrm{Gr}(d, D) = \{[X] | X \in \mathbb{R}^{D\times d}, X'X = I_d\}, \tag{2.7}$$

where $[X]$ represents a d-dimensional subspace which is generated by the columns of X. Since the unit orthogonal basis is not unique for a d-dimensional subspace, the point in $\mathrm{Gr}(d, D)$ might have different representations, i.e., suppose $[X], [Y] \in \mathrm{Gr}(d, D)$, $[X]$ and $[Y]$ are equivalent means that there exists $A \in \mathbb{R}^{d\times d}$ such that $X = YA$.

Although the definition of the Grassmannian manifold is different of any subset of the Euclidean space, it is proved that $\mathrm{Gr}(d, D)$ is equivalent to a quotient manifold $\mathbb{R}_*^{D\times d}/\sim$ (Absil *et al.*, 2009), where $\mathbb{R}_*^{D\times d}$ is an open subset of $\mathbb{R}^{Dd}$ and $\sim$ is some equivalence relationship which is used to induce the quotient manifold. Thus, $\mathrm{Gr}(d, D)$ has a natural manifold structure that is induced from Euclidean space.

With the manifold structure induced from $\mathbb{R}^{Dd}$, the related manifold concepts including tangent space, geodesic curve, exponential map $\exp$ and the logarithmic map $\log$ have analytic expressions, which enables $\mathrm{Gr}(d, D)$ to be used in practice. For example, if video sequences are viewed as points in a Grassmannian manifold, the geodesic distance can be used to measure the distance between two video sequences.

Suppose $[X] \in \mathrm{Gr}(d, D)$, the tangent space at $[X]$ is:

$$T_{[X]}\,\mathrm{Gr}(d, D) = \{X_\perp A | A \in \mathbb{R}^{(D-d)\times d}\}, \tag{2.8}$$

where $X_\perp \in \mathbb{R}^{(D-d)\times d}$ is the orthogonal complement of X. The Riemannian metric (canonical metric) on $\mathrm{Gr}(d, D)$ is induced from the vector inner product:

$$\begin{aligned}\langle Y_1, Y_2\rangle_g =&\langle \mathrm{Vec}(Y_1), \mathrm{Vec}(Y_2)\rangle \\ =& \, tr(Y_1 Y_2').\end{aligned} \tag{2.9}$$

for $\forall Y_1, Y_2 \in T_{[X]}\,\mathrm{Gr}(d, D)$.

Given a tangent vector $Y \in T_{[X]}\,\mathrm{Gr}(d, D)$, there is a unique geodesic curve f such that $f(0) = [X]$, and $\dot{f}(0) = Y$. Furthermore, the geodesic curve has analytic expression:

$$f(t) = [XV\cos(\Sigma t) + U\sin(\Sigma t)], \tag{2.10}$$

where $U \in \mathbb{R}^{(D-d)\times d}, \Sigma \in \mathbb{R}^{d\times d}, V \in \mathbb{R}^{d\times d}$ are obtained from the singular value decomposition (*SVD*) decomposition of $Y = U\Sigma V'$.

From the relationship between tangent vector and geodesic curve, the exponential map is

$$\exp_{[X]}(Y) = f(1) = [XV\cos(\Sigma) + U\sin(\Sigma)]. \tag{2.11}$$

For all $[X_2] \in \mathrm{Gr}(d, D)$, the logarithmic map $\log_{[X]}([X_2])$ is calculated by

$$\log_{[X]}([X_2]) = U_1\theta V_1', \tag{2.12}$$

where $U_1 \in \mathbb{R}^{D\times d}, V_1 \in \mathbb{R}^{d\times d}$ and $\theta = \arctan(\Sigma_1)$ are obtained from the *SVD* decomposition of $X_\perp X_\perp' X_2 (X'X_2)^{-1} = U_1\Sigma_1 V_1'$.

From Equation (2.9) and Equation (2.12), we have a natural way to define the geodesic distance, which is induced by the canonical metric, between $[X]$ and $[X_2]$:

$$d([X], [X_2]) = ||\log_{[X]}([X_2])||_F. \tag{2.13}$$

2.3 Manifold learning methods

Manifold learning has become an active research topic since the seminal works of *LLE* (Roweis and Saul, 2000) and Isomap (Tenenbaum *et al.*, 2000). In this section, different manifold learning techniques have been summarised in order to have a better understanding of how the geometry of the data set is explored. According to whether it extracts information only from a local region of the data set, we summarise the techniques in manifold learning into two categories. The category of preserving local similarities focus on exploring information between data points in their neighbourhood, and the category of preserving global similarities focus on preserving information which is extracted from the global view of the manifold space.

2.3.1 Preserving local geometric information

For preserving local similarities, the local region of data point x is often obtained by:

- The K nearest neighbours, i.e., the K data points $x_1, x_2, \cdots, x_K$ such that they have the smallest distances to x.
- An ϵ ball round x, i.e., all of the data points x_i and $||x - x_i||^2 < \epsilon$ where ϵ is a predefined constant.

Different manifold learning techniques explore the local geometric information from different perspectives. Then this local geometric information is used to learn a set of low dimensional points representing the data which preserve the geometric information of the original data set. In this category, we summarise four manifold learning techniques: *LLE*, *LE*, *LSTA* and *HLLE*.

LLE

In Locally Linear Embedding (*LLE*) (Roweis and Saul, 2000), the geometric information of data points in the local region is characterised by the reconstruction property between data points. For data point x_i, denote $x_{i_1}, x_{i_2}, \cdots, x_{i_K}$ as its local neighbours. Assuming the data points are distributed around some implicit manifold, we expect some linear patch structure among these $K+1$ data points. Thus, the x_i can be roughly reconstructed by its K neighbourhood points, and the weights are obtained by solving following optimisation problem:

$$\begin{gathered}\arg\min_{w_i} ||x_i - \sum_{j=1}^{K} w_{ii_j} x_{i_j}||^2 \\ s.t., w_{ii_1} + w_{ii_2} + \cdots + w_{ii_K} = 1.\end{gathered} \tag{2.14}$$

where the constraint $\sum_{j=1}^{K} w_{ii_j} = 1$ ensures that the reconstruction weights are not affected by translation of the data set.

Denote $W \in \mathbb{R}^{N\times N}$ as the learned weight matrix, where w_{ij} is its element in the i^{th} row and j^{th} column. The i^{th} row of W is the reconstruction weight for x_i (The weights for the data points which are not used for reconstructing x_i are set to 0).

With the information summarised in W, *LLE* aims to learn a set of data points $y_1, y_2, \cdots, y_N$ in a low dimensional space ($\mathbb{R}^d$), such that the reconstruction relationship (W) is preserved among these new data points. Denote

$$J(Y) = \sum_{i=1}^{N} ||y_i - \sum_{j=1}^{N} w_{ij} y_j||^2, \tag{2.15}$$

is the reconstruction error for $Y = [y_1, y_2, \cdots, y_N] \in \mathbb{R}^{d \times N}$.

From Equation (2.15), we can see that any translation to Y will not affect the final cost. Besides, the cost, $J(Y)$, equals to 0 when all y_i's are set to 0, so we need some normalisation. Thus two constraints

$$\sum_{i=1}^{N} y_i = 0, \tag{2.16}$$

and

$$\frac{1}{N} \sum_{i=1}^{N} y_i y_i' = I, \tag{2.17}$$

are introduced.

The final optimisation problem has an analytic solution. Denote $M = (I - W)'(I - W)$, $v_1, v_2, \cdots, v_d \in \mathbb{R}^N$ are its eigenvectors corresponding to the second smallest to the $(d+1)^{\text{th}}$ smallest eigenvalue. The column vectors of $V = [v_1, v_2, \cdots, v_d]' \in \mathbb{R}^{d \times N}$ are the corresponding vectors $y_1, y_2, \cdots, y_N$.

LE

In Laplacian Eigenmaps (*LE*) (Belkin and Niyogi, 2003), the similarity information between data points in the local region is represented by a similarity matrix $W \in \mathbb{R}^{N \times N}$. For two data points x_i and x_j in the same local region, the similarity between them can be calculated by the heat kernel, i.e.,

$$w_{ij} = \exp(-\frac{||x_i - x_j||^2}{t}), \tag{2.18}$$

where t is a predefined constant, otherwise, the similarity is 0. Alternatively, the similarity can be assigned 1 or 0 according to whether they belong to the same local region or not.

The similarity matrix preserves the relative distance information of the original data set. For example, higher similarity means that the corresponding data points are closer. This information is used to learn a set of low dimensional points $y_1, y_2, \cdots, y_N \in \mathbb{R}^d$. Belkin and Niyogi (2003) proposed to minimise the cost:

$$J(Y) = \sum_{i=1}^{N} \sum_{j=1}^{N} ||y_i - y_j||^2 w_{ij}. \tag{2.19}$$

The intuition behind the cost function is that if x_i and x_j are similar, then assign large weight for the corresponding distance of the points in the low dimensional space. This is equivalent to the effect that, if $||x_i - x_j||^2 > ||x_i - x_k||^2$, we expect to have the same relationship for low dimensional embeddings, i.e., $||y_i - y_j||^2 > ||y_i - y_k||^2$.

Denote $D \in \mathbb{R}^{N \times N}$ as a diagonal matrix where D_{ii} is the sum of the i^{th} row of W, and L is calculated by $L = D - W$, it is easy to verify that

$$J(Y) = Y'LY. \tag{2.20}$$

In order to avoid the degenerate solution and an arbitrary scale factor, two constraints are introduced:

$$Y'DY = I, \tag{2.21}$$

$$Y'D\mathbf{1} = \mathbf{0}. \tag{2.22}$$

The optimisation problem then has an analytic solution. For the generalised eigenvector problem

$$Lv = \lambda v, \tag{2.23}$$

denote $v_1, v_2, \cdots, v_d$ as the eigenvectors corresponding to the smallest non-zero d eigenvalues. The column vectors of $V = [v_1, v_2, \cdots, v_d]' \in \mathbb{R}^{d \times N}$ are the corresponding vectors of $y_1, y_2, \cdots, y_N$.

LTSA

In Local Tangent Space Alignment (*LTSA*) (Zhang and Zha, 2004), each local region is approximated by a linear patch, and then these patches are aligned with each other in the low dimensional space. Suppose $y_1, y_2, \cdots, y_N \in \mathbb{R}^d$, and a map f exists such that $x_i = f(y_i) + \varepsilon_i \in \mathbb{R}^D$ here x_i is viewed as an sample with noise ε_i.

With the assumption that data points in the local region have low intrinsic dimension, i.e., it can be approximated by a d-dimensional affine subspace, the data points in the local region are represented by a new coordinate corresponding to the local linear patch. The corresponding optimisation problem is

$$\min \sum_{j=1}^{K} ||x_{i_j} - (x + Q\theta_j)||_2^2, \tag{2.24}$$

where the columns of $Q \in \mathbb{R}^{D \times d}$ are a set of basis set of the corresponding subspace, $\Theta = [\theta_1, \theta_2, \cdots, \theta_K]$ is the local coordinates of the data points in the local region. The optimal solution for Equation (2.24) is obtained by assigning x as the mean of $X_i = [x_{i_1}, x_{i_2}, \cdots, x_{i_K}]$,

Q as $Q_i = [v_1, v_2, \cdots, v_d]$ where $v_1, v_2, \cdots, v_d$ are the leading d left singular vectors of $X_i(1 - \mathbf{11}'/K)$, and Θ as $\Theta_i = Q_i' X_i(1 - \mathbf{11}'/K)$.

Suppose the local coordinates are aligned in the low dimensional space, denote $Y = [y_1, y_2, \cdots, y_N]$ as the embedding data points, $Y_i = [y_{i_1}, y_{i_2}, \cdots, y_{i_K}]$ as the corresponding local region of X_i and $\overline{Y_i}$ as the mean of Y_i. Since Θ_i is a d dimensional approximation of X_i, we expect that there is an affine transform between Y_i and Θ_i, i.e.,

$$y_{i_j} = \overline{Y_i} + L_i\theta_j + \epsilon_j. \tag{2.25}$$

Denote $E_i = [\epsilon_1, \epsilon_2, \cdots, \epsilon_K]$ as the error matrix of the reconstruction which should be minimised. Given Y_i and X_i, the optimal solution for L_i, which gives minimal reconstruction error, is $Y_i(1 - \mathbf{11}'/K)\Theta_i^+$.

Considering the overall reconstruction error:

$$\begin{aligned}\sum_{i=1}^{N} ||E_i||_F^2 &= \sum_{i=1}^{N} ||Y_i(I - \mathbf{11}'/K) - L_l\Theta_i||_F^2 \\ &= \sum_{i=1}^{N} ||Y_i(I - \mathbf{11}'/K)(I - \Theta_i^+\Theta_i)||_F^2 \\ &= \sum_{i=1}^{N} ||YS_iW_i||_F^2 \\ &= ||YSW||_F^2,\end{aligned} \tag{2.26}$$

where S_i is the selection matrix such that $YS_i = Y_i$, $W_i = (I - \mathbf{11}'/K)(I - \Theta_i^+\Theta_i)$, $S = [S_1, S_2, \cdots, S_N]$, $W = diag(W_1, , W_2, \cdots, W_N)$ and the $|| \cdot ||_F$ is the Frobenius norm of a matrix.

With the constraint that $YY' = I$, the optimal solution is obtained by $Y = [v_1, v_2, \cdots, v_d]' \in \mathbb{R}^{d \times N}$, where $v_1, v_2, \cdots, v_d$ are its eigenvectors corresponding to the second smallest to the $(d+1)^{\text{th}}$ smallest eigenvalues of $SWW'S'$.

HLLE

In Hessian-based Locally Linear Embedding (*HLLE*) (Donoho and Grimes, 2003), the mapping f between manifold $\mathcal{M}$ and low dimensional embedding is assumed to be isometric, i.e., the geodesic distance between two data points in $\mathcal{M}$ is preserved after the mapping. Considering any coordinate component function $f : \mathcal{M} \to \mathbb{R}$, given $m \in \mathcal{M}$, suppose $g : U \to \mathbb{R}$ is induced by f, where $U \subset T_mM$ is an open set containing 0, and for any m' from the neighbourhood of m, it can be uniquely approximated by a data point $x \in T_mM$, thus $g(x)$ is defined as $f(m')$. The curviness of f at m is defined by

$$H_f^{tan}(m) = \left(\left. \frac{\partial^2 g(x)}{\partial u_i \partial u_j} \right|_{x=0} \right), \tag{2.27}$$

and the overall curviness is measured by

$$H(f) = \int ||H_f^{tan}(m)||_F^2 dm. \tag{2.28}$$

For estimating the Hessian matrix, Donoho and Grimes (2003) proved that Equation (2.28) is approximated by solving a least-square problem. Denote $X_i = [x_{i_1}, x_{i_2}, \cdots, x_{i_K}]$ as the K nearest neighbourhood data point of x_i and $Y_i = [f_{i_1}, f_{i_2}, \cdots, f_{i_K}]$. The entries in $\left(\left. \frac{\partial^2 g(x_i)}{\partial u_i \partial u_j} \right|_{x=0} \right)$ are estimated by least-square estimation, i.e., the *SVD* is used to obtain a basis for the tangent space $T_{x_i}(\mathcal{M})$, and the corresponding coordinates $U \in \mathbb{R}^{K \times d}$ of the neighbourhood data points. Denote $A_i \in \mathbb{R}^{K \times (d+1)}$, $B_i \in \mathbb{R}^{K \times d(d+1)/2}$ such that:

$$[A_i, B_i] = [1, U_{:,1}, U_{:,2}, \cdots, U_{:,d}, U_{:,1} \cdot U_{:,1}, U_{:,1} \cdot U_{:,2}, \cdots, U_{:,d} \cdot U_{:,d}], \tag{2.29}$$

where the $U_{:,i} \cdot U_{:,j}$ is the element-wise product of the i^{th} and j^{th} columns of U. Denote H_i as the remainder of B_i after the projection of A_i is removed, this can be done by Gram-Schmidt orthonormalisation on $[A_i, B_i]$, and the final $d(d+1)/2$ columns is H_i. Thus, the $||H_f^{tan}(x_i)||_F^2$ is approximated by

$$||H_f^{tan}(x_i)||_F^2 = Y_i' H_i H_i' Y_i', \tag{2.30}$$

and $H(f)$ is approximated by its discrete version:

$$\begin{aligned} H(f) &= \sum_{i+1}^{N} Y_i' H_i H_i' Y_i' \\ &= \sum_{i+1}^{N} Y S_i H_i H_i' S_i' Y \\ &= Y S H S' Y', \end{aligned} \tag{2.31}$$

where S_i is the selection matrix such that $YS_i = Y_i$, $S = [S_1, S_2, \cdots, S_N]$, and $H = diag(H_1 H_1', H_2, H_2', \cdots, H_N H_N')$.

For the final low dimensional embedding, with the constraint that $YY' = I$, the optimal solution is obtained by $Y = [v_1, v_2, \cdots, v_d]' \in \mathbb{R}^{d \times N}$, where $v_1, v_2, \cdots, v_d$ are its eigenvectors corresponding to the second smallest to the $(d+1)^{\text{th}}$ smallest eigenvalues of SHS'.

2.3.2 Preserving global geometric information

For preserving global geometric information, the optimisation focuses on the information obtained in a larger region rather than based on the information from the local region. Three manifold learning techniques are summarised as examples in this category. Tenenbaum *et al.* (2000) proposed *Isomap* to approximate the geodesic distances between any two data points, and then this information is preserved in learning the low dimensional embedding data points. Weinberger, Sha, and Saul (2004) proposed Maximum Variance Unfolding (*MVU*) to maximise overall distances between the embedding data points with the constraint that the distances of data points in the local region are preserved. Lin, He, Zhang, and Ji (2013) proposed to use a parallel field structure to learn the low dimensional embedding.

Isomap

In *Isomap* (Tenenbaum *et al.*, 2000), the geodesic distance between any two data points from the data set is estimated, and then Multidimensional Scaling (*MDS*) (Cox and Cox, 1994) is used to learn a low dimensional embedding.

For any two data points x_1 and x_2, the geodesic distance between them is the smallest curve length between the two data points along the manifold space. Tenenbaum *et al.* (2000) proposed to approximate the implicit manifold by a KNN graph of the data set. Each data point from the data set is viewed as a node in the graph, and then this data point is connected to its K nearest neighbourhood data points. The weights between connected data points are assigned by the corresponding Euclidean distances. With appropriate manifold assumptions, the geodesic distances of the data points in the same local neighbourhood are approximated with the corresponding Euclidean distances. For data points that are not in the same local neighbourhood, the geodesic distances are approximated by the shortest path searching through the KNN graph. In this way, the distance matrix $D \in \mathbb{R}^{N\times N}$ has better semantic information than the distance calculated by the Euclidean distances directly.

With the distance matrix D, a low dimensional embedding is obtained by the classical *MDS* method which aims to find a set of points such that the Euclidean distances between them are the same as D. Suppose $y_1, y_2, \cdots, y_N \in \mathbb{R}^d$ are the corresponding low dimensional embedding. Denote $H = I - \frac{1}{N}\mathbf{1}\mathbf{1}'$, thus $\tau(D) = \frac{1}{2}HDH'$ corresponds to the inner product of the data set after centring. The optimal solution is obtained by calculating the eigenvectors $v_1, v_2, \cdots, v_d$ which correspond to the largest d eigenvalues of $\tau(D)$. The column vectors of $V = [v_1, v_2, \cdots, v_d]' \in \mathbb{R}^{d\times N}$ are $y_1, y_2, \cdots, y_N$.

MVU

In order to unfold the manifold in low dimensional space, Weinberger *et al.* (2004) proposed Maximum Variance Unfolding (*MVU*) to maximise the distribution of the embedding data points, i.e. the optimisation objective is to maximise

$$\sum_{i=1}^{N}\sum_{j=1}^{N} ||y_i - y_j||^2. \tag{2.32}$$

Imagine that the data points are sampled from a piece of paper which is folded in 3D space. Now, we are doing some operations on this paper such that the data points from the paper are as far apart as possible. After some number of operations, the paper will be flattened. Bengio, Paiement, Vincent, Delalleau, Roux, and Ouimet (2004) proved that kernel principal component analysis (*KPCA*) has an intimate relationship with manifold learning techniques including *Isopmap*, *LLE* and *LE*. For example, different manifold learning methods are viewed as special Kernel *PCA* if the kernel function is defined appropriately. This motivates *MVU* to model the relationship of data points in the kernel space, and the optimisation objective is to find the specific kernel matrix with some appropriate constraints.

Suppose ϕ maps each data point x into a kernel space, the first constraint of the points in the kernel space is that they should be centred, i.e., $\sum_{i=1}^{N}\phi(x_i) = 0$. Considering the kernel matrix $K \in \mathbb{R}^{N\times N}$, where k_{ij} is the inner product between $\phi(x_i)$ and $\phi(x_j)$. Thus the centring condition is equivalent to:

$$0 = |\sum_{i=1}^{N}\phi(x_i)|^2 = \sum_{i=1}^{N}\sum_{j=1}^{N}\phi(x_i)'\phi(x_j) = \sum_{i=1}^{N}\sum_{j=1}^{N} k_{ij}. \tag{2.33}$$

For data points in the same local neighbourhood, their distance should be preserved, i.e.,

$$|x_i - x_j|^2 = |\phi(x_i) - \phi(x_j)|^2. \tag{2.34}$$

Denote $g_{ij} = x_i' x_j$, Equation (2.34) is equivalent to:

$$g_{ii} + g_{jj} - 2g_{ij} = k_{ii} + k_{jj} - 2k_{ij}. \tag{2.35}$$

The optimisation aims to maximise the overall distances between $\phi(x_1)$, $\phi(x_2), \cdots$, $\phi(x_D)$. Since

$$\frac{1}{2N} = \sum_{i=1}^{N}\sum_{j=1}^{N} |\phi(x_i) - \phi(x_j)|^2 = \mathrm{Tr}(K), \tag{2.36}$$

the final optimisation problem is to find a kernel matrix which is positive semidefinite, thus it is a semidefinite programming problem with two constraints.

With the solution K obtained by semidefinite programming, the d dimensional embedding is obtained by calculating the eigenvectors $v_1, v_2, \cdots, v_d$ which correspond to the largest d eigenvalues of K. The column vectors of $V = [v_1, v_2, \cdots, v_d]' \in \mathbb{R}^{d \times N}$ are $y_1, y_2, \cdots, y_N$.

PFE

In parallel field embedding (*PFE*) (Lin *et al.*, 2013), a parallel field structure is learned from the data set, and then this information is used to learn the low dimensional embedding. A vector field ξ on the manifold $\mathcal{M}$ is a smooth map such that $\xi(x) \in T_x(\mathcal{M})$, for all $x \in \mathcal{M}$. Intuitively, it selects one tangent vector in each tangent space $T_x(\mathcal{M})$. A parallel vector field is a generalisation of parallel vectors in manifold space. It is formally defined as a vector field satisfying $\bigtriangledown \xi = 0$, where $\bigtriangledown$ is the covariant derivative on the manifold (Lin *et al.*, 2013).

The optimisation problem for learning the vector field ξ is:

$$\begin{aligned} \min J(V) &= \int_M || \bigtriangledown V||_F^2 dx \\ s.t., &\int_M ||V||^2 = 1. \end{aligned} \tag{2.37}$$

Denote $T_i \in \mathbb{R}^{D \times d}$ as the matrix representing a basis of $T_{x_i}(\mathcal{M})$, thus each tangent vector $\xi(x_i) \in T_{x_i}(\mathcal{M})$ can be represented as $T_i v_i$. the discrete version of Equation (2.37) is approximated by

$$\sum_{i=1}^{N} \sum_{j=1}^{N} w_{ij} ||P_i T_j v_j - T_i v_i||, \tag{2.38}$$

where the weight is defined as $w_{ij} = \frac{1}{||x_i - x_j||^2}$ if x_i and x_j are in the same local neighbourhood, and $w_{ij} = 0$ otherwise, $P_i = T_i T_i'$ is the orthogonal projection from $\mathbb{R}^D$ to the tangent space $T_{x_i}(\mathcal{M})$. With the discrete version of Equation (2.37), the discrete version of V, i.e., its value at the position of each data point, is obtained by solving the corresponding eigenvector problem.

Suppose f is a map from $\mathcal{M}$ to an open subset of $\mathbb{R}^d$ satisfying $f(x_i) = y_i$. The equation $\bigtriangledown f = V$, where $\bigtriangledown f$ is the gradient field of f, ensures that f is a linear function which changes linearly along the geodesic curve on the manifold. Intuitively, f behaves as unfolding the manifold into $\mathbb{R}^d$. The optimisation problem for learning this linear function f

is:

$$J(f) = \int_{\mathcal{M}} || \bigtriangledown f - V||^2 dx. \tag{2.39}$$

The discrete version of Equation (2.39) is approximated by

$$J(Y) = \sum_{i=1}^{N} \sum_{j=1}^{N} w_{ij} ||(P_i(x_i - x_i))' V_{x_i} - y_j + y_i||^2. \tag{2.40}$$

Thus, the final solution is solved by setting $\frac{\partial J(Y)}{\partial Y} = 0$.

In this section, we have summarised some examples of how geometric information from a data set is mined and then used for learning low dimensional embeddings. The summarised techniques have various applications in related fields. For example, *LE* has been developed into a linear manifold learning method (He and Niyogi, 2004) and a regulariser for regression problems (Cai, He, Zhang, and Han, 2007). Our focus is how the geometric information is explored and used for learning the low dimensional embedding. Thus, we have not summarised other techniques in manifold learning including the unified viewpoint of different manifold learning techniques (Bengio *et al.*, 2004; Lawrence, 2004), learning the low dimensional embedding through probability distribution (Hinton and Roweis, 2002; Van Der Maaten and Hinton, 2008) and label information (Yan, Xu, Zhang, Zhang, Yang, and Lin, 2007).

2.4 Summary

This chapter provides the theoretic background for the work in this thesis, and the summarised manifold learning techniques also shed light on related works in later chapters. For example, our work in video classification heavily relies on the Grassmannian manifold and its related computations; the tangent space is used to approximate the distribution of data points in local regions and this kind of information is used to learn the encoding functions for ANN search tasks; the techniques in manifold learning provide examples as well as motivate us to mine geometric information for appropriate computer vision problems.

Part I

Exploring geometric structures for clustering problems

Chapter 3

Improved Spectral Clustering using Adaptive Mahalanobis Distance

Note: Some portions of this chapter are taken from Fu *et al.* (2013).

In this chapter, we address the clustering problem when data points belong to different manifolds that are close to or intersect with each other. Traditional spectral clustering algorithms usually fail to separate such manifolds. We propose to improve the similarity matrix construction step in spectral clustering by learning a local Mahalanobis distance. We show the effectiveness of the method on some artificial data sets, and also incorporate this modification into recent related algorithms, and compare these algorithms on some real data sets.

3.1 Introduction

In the past few years, a lot of work has been done to extend clustering algorithms to ever more difficult problems. Specifically, algorithms have been designed to cluster data which are sampled from multiple manifolds. These manifolds may be very close to each other and may even intersect. For this kind of clustering problem, we need to construct an elaborate similarity matrix W to group the data using spectral clustering. Chen and Lerman (2009) have constructed a similarity matrix based on the polar $\sin$e, which is a high dimensional generalisation of the $\sin$e function. For each data point, the polar $\sin$e is estimated based on data points randomly chosen from the data set. Thus, it is a global algorithm and has good performance when the data points are sampled from linear manifolds.

The similarity matrix W can also be estimated based on information extracted from local regions. Wang, Jiang, Wu, and Zhou (2011) incorporated tangent space information into the similarity matrix. They used the mixture of probabilistic principal component analysers (*MPPCA*) (Tipping and Bishop, 1999) model to fit the data set. In this model each data point has a corresponding tangent space which is learned from *MPPCA*. With this information, the similarity matrix is constructed from the distance information and the angle information of corresponding tangent spaces. Gong, Zhao, and Medioni (2012) estimated the local tangent space using a weighted low-rank matrix factorisation. The main assumption is that, when calculating the tangent space at x, the more distant points from the neighbourhood contribute more to the error than nearby ones do. Thus they introduce a penalty for neighbourhood points according to their distances to x. With the tangent space information for each data point, they construct a similarity matrix W using both distance information and angle information of the local tangent spaces. Finally, Arias-Castro, Lerman, and Zhang (2013) proposed three algorithms which address the manifold intersection problem. The central idea behind these algorithms is to incorporate local covariance information which is calculated by each x and its neighbourhood points to construct a similarity matrix.

In this work, we propose an algorithm designed to improve the selection of neighbourhoods in the case of data sampled from multiple neighbourhoods. By improving neighbourhood selection, we improve the similarity matrix used by spectral clustering algorithms. Like previous works (Gong *et al.*, 2012; Wang *et al.*, 2011; Arias-Castro *et al.*, 2013), we are trying to construct a better similarity matrix for spectral clustering. Unlike previous work, we do not explicitly estimate local tangent spaces, nor do we use a single covariance measurement to reject certain edges. Instead, we select edges by employing an iterative Mahalanobis distance calculation. Since we are concerned only with neighbourhood selection, the modified neighbourhood selection method can be applied as a pre-processing step for various spectral clustering algorithms (Gong *et al.*, 2012; Wang *et al.*, 2011; Ng *et al.*, 2002).

In Section 3.2, we introduce the necessary background for the modified neighbourhood selection method, including details on the Mahalanobis distance and spectral clustering. In Section 3.3, we describe how to select neighbourhoods using an iterative computation of Mahalanobis distance. We provide examples and discuss computational complexity. In Section 3.4, we incorporate the modified neighbourhood selection method into other algorithms and compare the resulting performance on some real data. In Section 3.5, we summarise the work in this chapter.

3.2 Background

3.2.1 Mahalanobis distance

The modified neighbourhood selection method uses the Mahalanobis distance to select neighbourhoods. To define the Mahalanobis distance, we suppose that $x, y \in \mathbb{R}^D$ and that $\Sigma \in \mathbb{R}^{D\times D}$ is a symmetric positive definite covariance matrix. The Mahalanobis distance is defined as $d_\Sigma(x, y) = \left((x-y)'\,\Sigma^{-1}\,(x-y)\right)^{-\frac{1}{2}}$. Under the Mahalanobis distance, the space $\mathbb{R}^D$ can be viewed as normalised by Σ. In Fig. 3.1, we show a unit sphere in the Mahalanobis distance using two different covariance matrices.

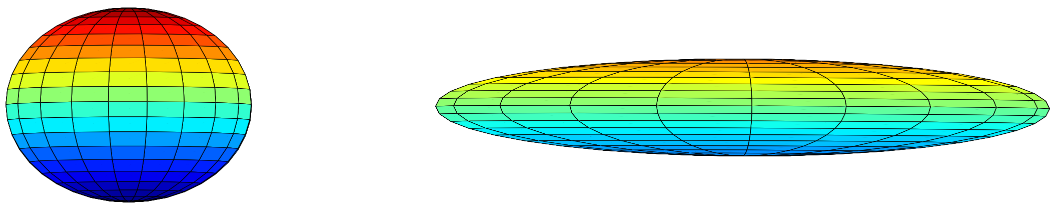


Fig. 3.1. Unit balls under different Mahalanobis distances. On the left we use $\Sigma = I$, and on the right we use a diagonal matrix with entries $(3, 1, 1)$ for Σ.

The Mahalanobis distance has been widely used for solving machine learning problems. For example, Goldberg, Zhu, Singh, Xu, and Nowak (2009) proposed the Multi-Manifold Semi-Supervised learning algorithm. Their main idea was to reduce the clustering size by partitioning the unlabelled data points into small regions. Next, for the size-reduced data set, the local covariance for each data point was calculated. The similarity matrix was then constructed by this Mahalanobis distance information and the Hellinger distance between them. Kushnir, Galun, and Brandt (2006) also proposed to use Mahalanobis distance calculated by covariance matrices around the local region, to find the structure of the data set. In this work, we propose to learn the Mahalanobis distance iteratively. The hope is that such an approach will allow a better determination of the neighbourhood of each data point.

Another closely related field is distance metric learning (Yang and Jin, 2006) which learns the Mahalanobis distance for the data set. Distance metric learning can be divided into two branches. One is supervised distance metric learning which means we have label information of the data points. With this kind of information, we can find the optimal Mahalanobis matrix for the distance measurement (Xing, Jordan, Russell, and Ng, 2002; Weinberger, Blitzer, and Saul, 2005). The other is unsupervised distance metric learning which is closely related to manifold learning or dimensionality reduction (Yang and Jin, 2006). It aims to find a low dimensional structure which is usually from one latent manifold. Notice that our modification is to learn the local Mahalanobis distance for the unsupervised clustering problem, and the data points are assumed to be sampled from multiple manifolds which are close to or intersect with each other.

3.2.2 Spectral clustering

Suppose we have a data set $X = \{x_1, x_2, \ldots, x_N\} \subset \mathbb{R}^D$. The first step in any spectral clustering algorithm is the construction of a weighted similarity graph. In this graph, vertices correspond to data points x_i and edges give the similarity between two points x_i and x_j. For example, we might form the weighted similarity graph using ϵ-balls to specify neighbourhoods: for each data point x_i, we connect it to point x_j if the Euclidean distance $d(x_i, x_j) \leq \epsilon$. Another common approach for generating weighted similarity graphs is to connect each data point to its K nearest neighbours (KNN graph). The similarity between the neighbourhood data points is assigned by Gaussian similarity function $w(x_i, x_j) = \exp(-\frac{||x_i - x_j||^2}{2\sigma^2})$.

After assigning an appropriate weight to each edge, we get the similarity matrix $W = (w_{ij})$ for the graph. Note that $w_{ij} = 0$ if an edge does not exist. Spectral clustering (Ng *et al.*, 2002) is done by calculating the normalised Laplacian $L = I - D^{-1/2}WD^{-1/2}$, where D is a diagonal matrix with $D_{ii} = \Sigma_{j=1}^{N} w_{ij}$. Next, we compute the smallest k eigenvalues of the eigenvalue problem $Lu = \lambda u$, where $u_1, \ldots, u_k$ are the corresponding eigenvectors. If we form a matrix $U = (u_1, u_2, \ldots, u_k) \in \mathbb{R}^{N \times k}$, $T = (y'_1, y'_2, \ldots, y'_N)' \in \mathbb{R}^{N \times k}$ is obtained by normalising each row of U, then y_i is viewed as a representation of x_i. Finally, we cluster $\{y_1, y_2, \ldots, y_N\}$ into k clusters using K-means. An overview of the process is given in Algorithm 1.

The similarity matrix W plays a pivotal role in spectral clustering. The values in the similarity matrix are used to reflect the relationship between data points, i.e., high similarity between data points means they are close to each other and from the same class. In the extreme case, if each data point only connects data points from the same class, then T only has k distinct rows and these distinct rows are orthogonal with each other (Ng *et al.*, 2002). Thus, the spectral clustering algorithm can cluster the data set successfully. In real-word data sets, the similarity matrix might not be ideal, but it has been shown that if the similarity matrix is close enough to the ideal case, the spectral clustering algorithm can cluster the data set successfully.

Algorithm 1 Spectral Clustering Algorithm (Ng *et al.*, 2002)

```
   Input: Data set $X \in \mathbb{R}^{D\times N}$, the number of clusters $k$.
   Output: A set of $k$ clusters.
1: Construct a similarity graph, denote the corresponding adjacency matrix as $W$.
2: Calculate the normalized Laplacian $L = I - D^{-1/2}WD^{-1/2}$, where $D$ is a diagonal matrix with $D_{ii} = \Sigma_{j=1}^{n} w_{ij}$
3: Calculate the smallest $k$ eigenvalues of the eigenvalue problem $Lu = \lambda u, u_1, u_2, \cdots, u_k$ are the corresponding eigenvectors.
4: Form $U = (u_1, u_2, \cdots, u_k) = (y_1', y_2', \cdots, y_N')' \in \mathbb{R}^{N\times k}$ and normalize each row of $U$, denote the new matrix as $T = (y_1', y_2', \ldots, y_N')' \in \mathbb{R}^{N\times k}$
5: Cluster $\{y_1, y_2, \cdots, y_N\}$ into $k$ clusters by $K$-means algorithm.
```

3.3 Algorithm

3.3.1 Motivation

From the discussion of spectral clustering, we can see that the clustering results are associated with the quality of the similarity matrix. When data points from different classes are close to each other, misconnections arise, i.e., we might assign large similarity weights between data points from different classes. This is the main cause of failure if we use spectral clustering to divide the data set. Thus our focus is to explore appropriate geometric information for learning the similarity matrix W.

We aim to extract some geometric information for deciding the neighbourhood of the data points. If there are fewer misconnections in the similarity graph, the performance of spectral clustering will improve. Clustering is an unsupervised learning problem, i.e., we do not have any prior information of the data distribution or class labels, so we use isotropic ball distance to find the nearest neighbourhood points. The distribution information of data points in a local region is used to learn the Mahalanobis distance information. Notice that the learned Mahalanobis distance can be used to find the nearest neighbour points again. We repeat the above procedure hoping to reduce the number of wrong connections.

An example is shown in Fig. 3.2. Here the data points are chosen from two intersecting lines with some random noise. For the blue point x_i, we select its 20 nearest neighbourhood points which are shown in the top-left position, we notice that there are at least 6 misconnections. Calculate the covariance of these $(20 + 1)$ points, and then use this covariance as the Mahalanobis distance to find its 20 nearest neighbourhood points. For this example,

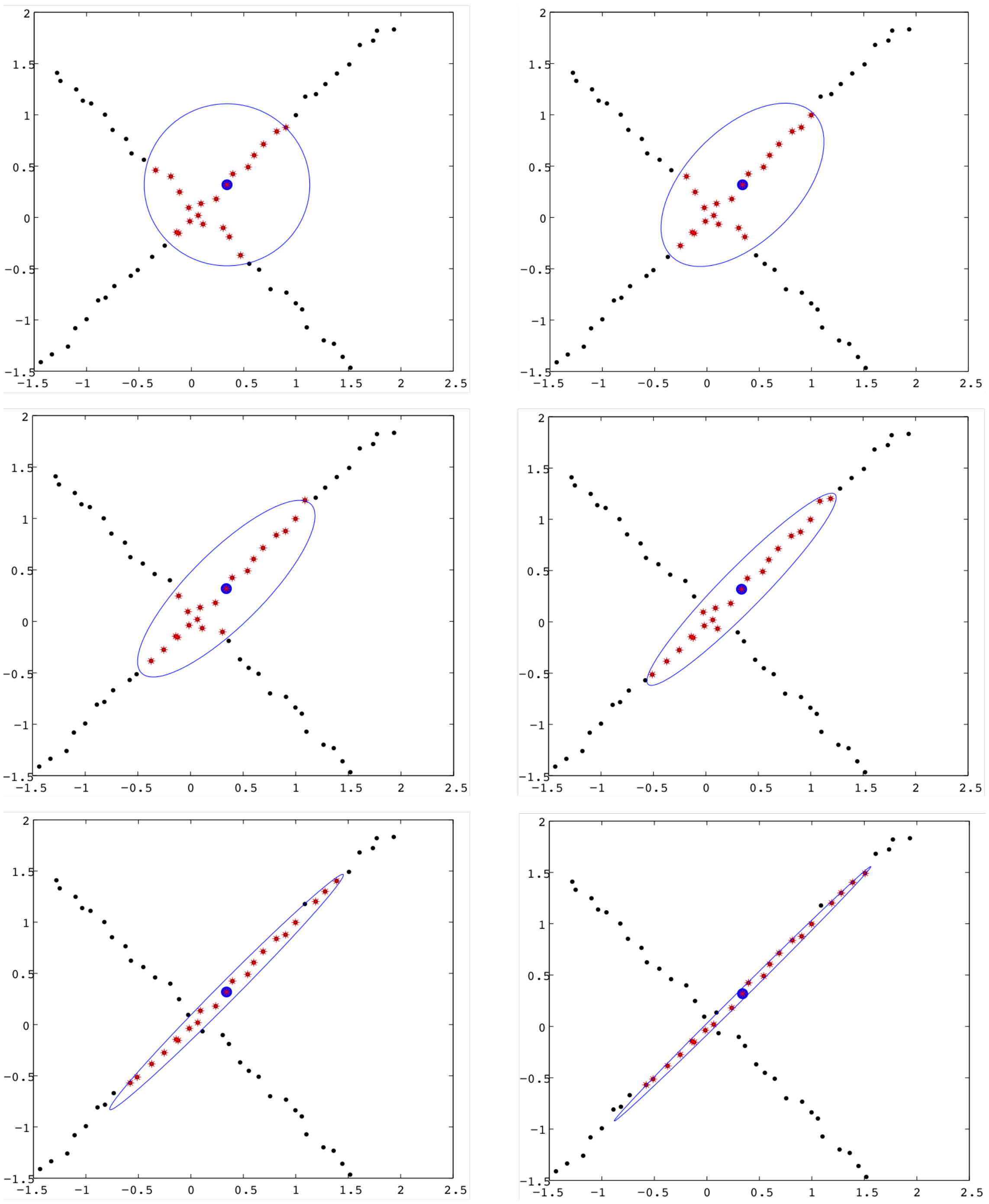


Fig. 3.2. Neighbourhood selection. Here we consider a data set with two intersecting lines and a neighbourhood centre near the intersection point. From the top left to the bottom right, we show snapshots of the neighbourhood of the blue data point in each iteration. The neighbourhood centre is shown enclosed within a circle surrounding the neighbourhood. Non-Neighbours are shown in grey. As the algorithm converges, the neighbourhood improves so that the intersecting line is ignored.

we see that after 6 iterations, the blue point has 'correct' connections.

3.3.2 Algorithm

The neighbourhood learning algorithm is summarised in Algorithm 2. This algorithm aims to select neighbourhoods respecting to the manifold structures, regardless of nearby manifolds or manifold intersections, and it is based on iteratively recomputing the Mahalanobis distance for a given neighbourhood centre. For i^{th} data point x_i, the main task is to identify its K nearest neighbours. In the absence of prior information, we assume an isotropic Mahalanobis distance ($\Sigma = I$) (Line 2) to find the closest K neighbours (Line 4). Using these neighbours, we compute the covariance and re-compute the Mahalanobis distance (Line 5). We return to Line 3 and select a new set of K neighbours based on the new Mahalanobis distance and repeat unless there is no change of the covariance or neighbourhood of the x_i (Line 6-10).

In practice, we do not wait for convergence, but terminate after a fixed number of iterations I_{max}. After the neighbourhood selection process, we apply a spectral clustering algorithm. We denote the resulting algorithm as *Modified-SC*.

Algorithm 2 Identify neighbourhood using Mahalanobis Distance

```
    Input: Data set $X$ with $N$ elements; neighbourhood size $K$; maximum number of iterations $I_{max}$.
    Output: Neighbourhood for each data point.
 1: for $i = 1$ to $N$ do
 2:     Let $\Sigma = I$.
 3:     for $j = 1$ to $I_{max}$ do
 4:         Use distance $(x_j - x_i)'\Sigma^{-1}(x_j - x_i)$ to find $K$ nearest neighbours of $x_i$.
 5:         Calculate the covariance Cov of $\{x_{i1}, x_{i2}, \ldots, x_{iK}\}$.
 6:         if Cov $= \Sigma$ then
 7:             Go to Line $14$.
 8:         else
 9:             Set $\Sigma =$ Cov, and return to Line $5$.
10:         end if
11:     end for
12:     Record the points $\{x_{i1}, x_{i2}, \cdots, x_{iK}\}$ as $x_i$'s neighbourhood.
13: end for
```

3.3.3 Computational complexity

Taking N to be the total number of points in the data set and K is the size of neighbourhood, spectral clustering takes $\mathcal{O}(KN^2)$ to find nearest neighbourhood points and solving a generalised eigenvalue problem takes $\mathcal{O}(N^3)$. Our modification involves identifying neighbourhood iteratively at most I_{max} times. The time required to identify neighbourhood is $\mathcal{O}(KN^2 I_{max})$. Since solving the generalised eigenvalue problem dominates the main computation, the modified algorithm has roughly the same complexity as the traditional spectral clustering algorithm. Besides, we should note that the time for identifying the neighbourhood in the proposed method can be further reduced by fast approximate nearest neighbour search methods (Bentley, 1975; Indyk and Motwani, 1998).

3.4 Experiments

3.4.1 Artificial examples

We examine some artificial examples which cannot be partitioned using traditional spectral clustering algorithms. The first example is two intersecting lines. For this example, we generated 400 points uniformly sampled from two lines. We used $K = 10$ nearest neighbours and identified $k = 2$ clusters. The results of a traditional spectral clustering algorithm (Ng *et al.*, 2002) compared with the same spectral clustering algorithm modified using our neighbourhood selection is shown in Fig. 3.3.

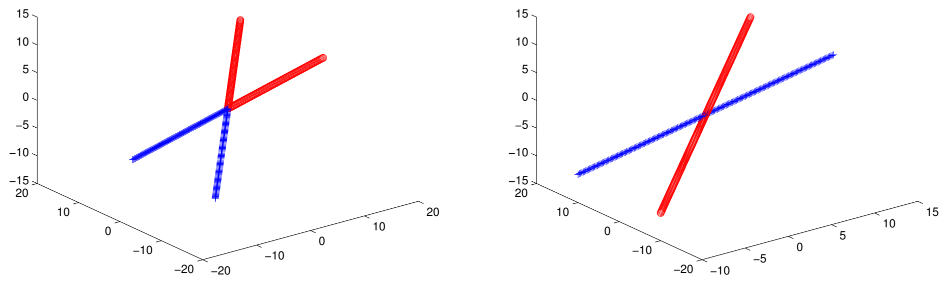
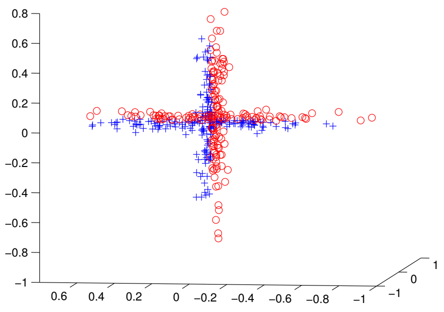


Fig. 3.3. Clustering for two intersecting lines. On the left we show the result of a traditional spectral clustering algorithm, and on the right the result of the same algorithm modified by first applying Algorithm 2.

In our next example, we use two intersecting planes. For this example, we generated

200 points sampled from a Gaussian distribution from each plane. We again used $K = 10$ nearest neighbours and identified $k = 2$ clusters. Clustering by our method yielded only two misclassified points. A comparison with the traditional method is shown in Fig. 3.4.



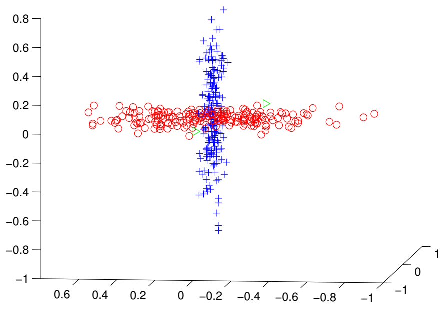


Fig. 3.4. Clustering for two intersecting planes. One plane is in the plane of the page, the other is orthogonal to the page and horizontal. On the left we show the results of a traditional spectral clustering algorithm, and on the right the result of the same algorithm modified by our neighbourhood selection scheme. There are two misclassified points using our approach, shown in green with triangular shape.

The previous two examples show that for manifold intersections, the performance of spectral clustering can be improved using our iterative algorithm for learning the Mahalanobis distance. We note that when the data lies on a manifold without error, the method will work for a large range of K. However, if there is noise, the value of K should be large enough to eliminate the noise. For example, in Fig. 3.5, we show that for two intersecting lines with noise, the data can still be partitioned successfully if we choose $K = 20$.

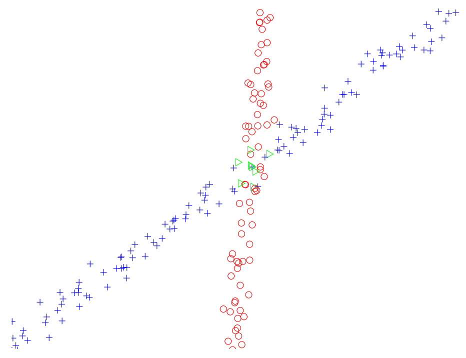


Fig. 3.5. Clustering for two intersecting lines with noise. The *Modified-SC* algorithm is able to cluster noisy data using larger neighbourhoods. Errors are shown in green with triangular shape.

3.4.2 Motion segmentation by trajectories

The Hopkins 155 motion segmentation database (`http://www.vision.jhu.edu/data/`) (Tron and Vidal, 2007) is a benchmark for testing motion segmentation and subspace learning algorithms. There are 155 data sets, including 13 articulated sequences which relate to people's motion, 38 traffic sequences, and 104 checkerboard sequences. The checkerboard sequences display different combinations of movements of both checkerboard and camera, such as rotation and translation. Some of the sample images are shown in Fig. 3.6.

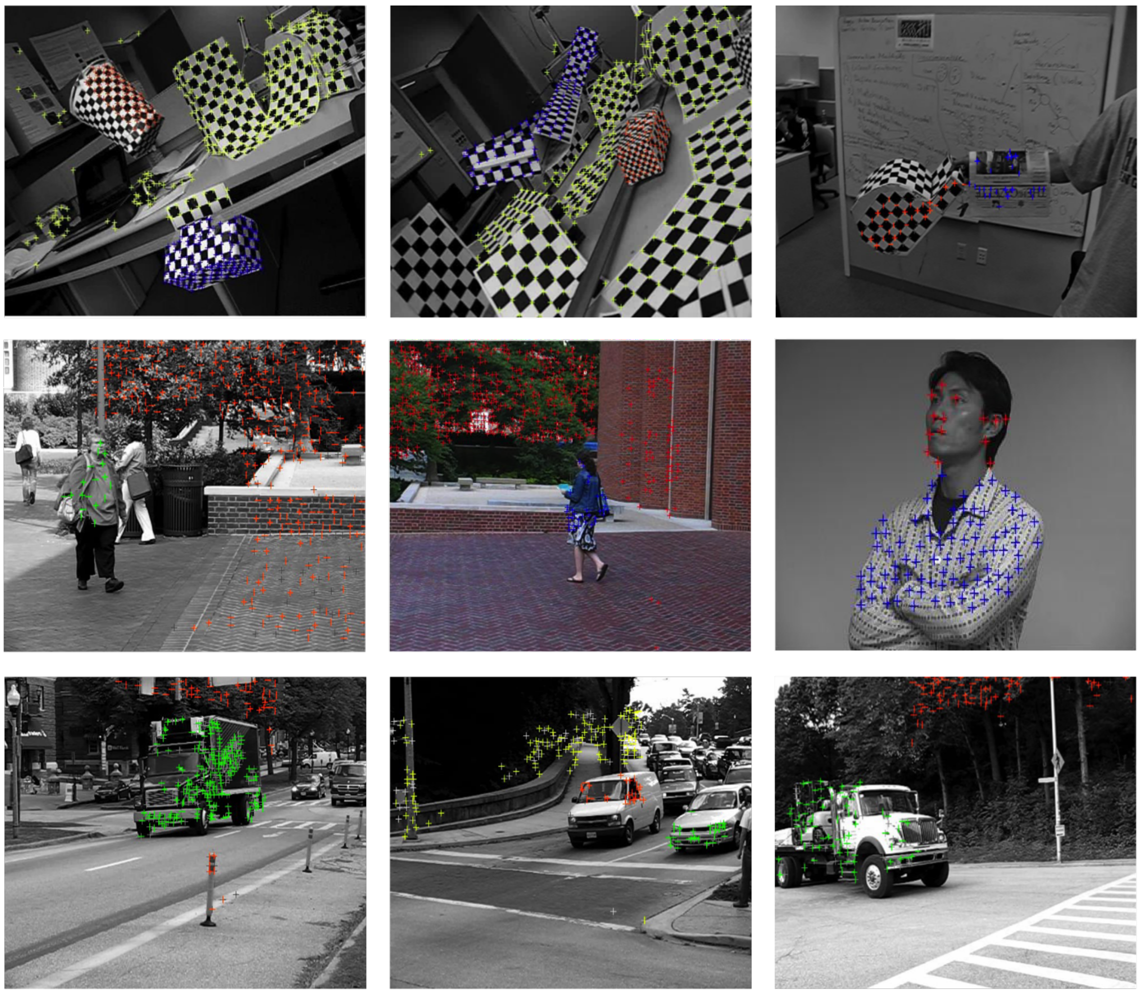


Fig. 3.6. Sample images from the Hopkins 155 motion segmentation database. Each image is a representative of one video sequence. The ground truth of the different segmentations is distinguished by different colors.

Suppose there are F frames in one video sequence. A specific feature can be characterised by its position (x_i, y_i) in the i^{th} frame. The trajectory of this feature is defined as

$(x_1, y_1, x_2, y_2, \ldots, x_F, y_F) \in \mathbb{R}^{2F}$. Typically trajectories belonging to a specific rigid movement lie in the same low dimensional manifold. Therefore, we can use multi-manifold learning algorithms to separate them.

We applied our neighbourhood connection method to the standard spectral clustering algorithm (*SC*), spectral multi-manifold clustering (*SMMC*) (Wang *et al.*, 2011), and robust multi-manifold structure learning (*RMMSL*) (Gong *et al.*, 2012). We name the modified algorithms *Modified-SC*, *Modified-SMMC* and *Modified-RMMSL*. We compare these modified algorithms with *SC*, *SMMC*, *RMMSL*, generalised principal component analysis (*GPCA*) (Vidal, Ma, and Sastry, 2005), and spectral curvature clustering (*SCC*) (Chen and Lerman, 2009) on the Hopkins 155 motion segmentation database. The code for *GPCA* (Vidal *et al.*, 2005), *SCC* (Chen and Lerman, 2009), and *SMMC* (Wang *et al.*, 2011) was downloaded from corresponding authors' homepages[1][2][3].

The results of the comparison are shown in Table 3.1. We show the performance of some specific data sets as well as the average performance across the whole database. Each experiment is repeated 100 times, and the misclassification rate is reported. To facilitate visualisation of the motion segmentation results, we present the segmentation results of the first three data sets in Fig. 3.7–3.8 respectively. The clustering method and the corresponding misclassification rate are shown at the top of each picture. We use the 10^{th} frame of the corresponding video sequence as a representative. The top left picture shows the positions of all the features. The remaining pictures show the segmentation results of the 8 clustering methods respectively. Features from different clusters are distinguished by different colours. In most cases, our modification improves the results of the standard method, and in certain cases the improvement is significant.

Note that some of these algorithms have already been tested on this data set, and the results may be different from those shown in Table 3.1. This is probably due to differences in parameter selection and data pre-processing used in each algorithm. For example, Wang *et al.* (2011) tested the *SCC* and *SMMC* algorithms without data pre-processing, i.e. on the $2F$ dimensional data directly, while Chen and Lerman (2009) pre-processed the data using *PCA*. Further, Gong *et al.* (2012) performed their experiments by first tuning the parameters, then choosing the best parameter for the trials, whereas we do not tune the parameters. Hence,

[1]`http://www.vision.jhu.edu/code.htm`
[2]`http://www.math.sjsu.edu/~gchen/scc.html`
[3]`http://cs.nju.edu.cn/zhouzh/zhouzh.files/publication/publication.htm`

	kanatani1	cars6	2T3RTCR	cars5_g13	1R2RCT_A	Average
GPCA	0%	0%	42.16%	1.17%	5.94%	10.21%
SC	0.74%	37.72%	1.31%	16.45%	3.80%	9.51%
Modified-SC	**0%**	**0%**	**0.75%**	16.55%	6.65%	**8.88%**
SMMC	3.26%	0.82%	15.17%	9.45%	15.18%	9.62%
Modified-SMMC	**1.92%**	**0.82%**	**14.52%**	**9.20%**	**14.66%**	**9.35%**
RMMSL	0%	8.19%	34.70%	6.38%	34.44%	8.55%
Modified-RMMSL	**0%**	**0%**	**1.31%**	**5.51%**	**4.51%**	**7.37%**
SCC	0%	19.36%	0.57%	0%	0.04%	3.40%

Tab. 3.1. Performance comparison. Here we compare a host of spectral clustering algorithms using the Hopkins 155 motion segmentation database. Misclassification rates are reported, and bold-face is used when our modification is as good or better than the result of the standard algorithm. In some cases there are significant improvements in sense of the misclassification rate.

the results shown in Table 3.1 can only be compared in the context of our trials. In other words, we tested the methods using the same parameters and pre-processing. We only varied the method of edge connection (neighbourhood selection).

Due to the large number of methods considered, however, there are some subtleties in our analysis, which we now discuss. First, the variety of algorithms requires a standard method of pre-processing and parameter selection which will work for all methods. For pre-processing, we first use *PCA* to project the trajectories into 5 dimensional space. Actually, the dimension of trajectories from a rigid-body movement ranges from 2 to 4 (Tron and Vidal, 2007). Thus after projection the trajectories are still on different low dimensional manifolds in $\mathcal{R}^5$. Besides, *GPCA* is designed to address the subspace intersection problem. Thus $D = 5$ is the minimum dimension for *GPCA*.

Some algorithms, such as *SMMC*, *Modified-SMMC*, *RMMSL*, *Modified-RMMSL*, and *SCC* need intrinsic dimension information, which we always set as $d = 3$ since this leads to a better estimation of the distribution of the trajectories. For *SC*, *Modified-SC*, *SMMC*, *Modified-SMMC*, *RMMSL* and *Modified-RMMSL*, the neighbourhood size K is set to $2\lceil \log(N) \rceil$, as suggested when using SMMC (Wang *et al.*, 2011). For *SMMC*, the number of components M of *MPPCA* and the adjustable parameter o are using the default settings, i.e., they are set to $\lceil \frac{N}{10d} \rceil$ and 8. Since the number of features in this data set ranges from 39 to 556, the number of components is up to 18. We found that the *MPPCA* used in *SMMC* sometimes

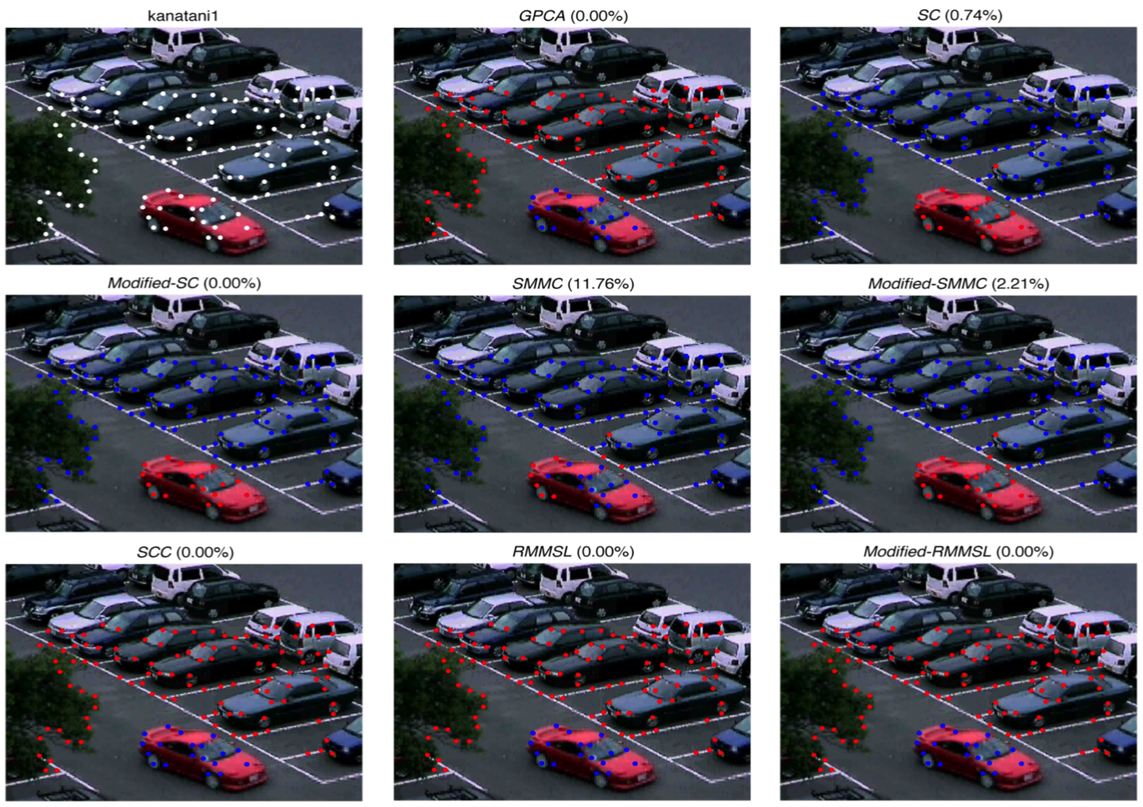


Fig. 3.7. Clustering results for kanatani1 video sequence. There are 30 frames in this data set. The red car is driving in the carpark, and the camera is moving in order to follow the car.

returns undesirable results. For example, when the number of components is set too large, some component in *MPPCA* might have only one data point and this leads to computation error when estimating the tangent information in the local region. Thus, the upper limit and lower limit of the number of components M are set to be 3 and 9 respectively.

When comparing *SC* and *Modified-SC*, the only difference is the parameter for the number of iterations to use in the Mahalanobis distance computation. For *SC*, we use a single iteration, and for *Modified-SC* we use 10 iterations. For the comparison of *SMMC* and *Modified-SMMC*, our results depend heavily on the outcome of the *MPPCA* process. Thus for comparison, we fix the *MPPCA* procedure. In other words, we use the same result from *MPPCA*, then run the different algorithms. For *RMMSL* and *Modified-RMMSL*, we note that the outliers have been previously removed in the Hopkins 155 data set. Thus we can skip the outlier detection process in *RMMSL*, so we set $\sigma_c = 0.02, \sigma_n = 1$, and $K_{th} = 5$ for the self-tuning step. After estimating the weighted tangent space for each data point, we apply the different edge connection methods to continue the experiment.

The results in Table 3.1 are produced using a set of fixed parameters. For real world data

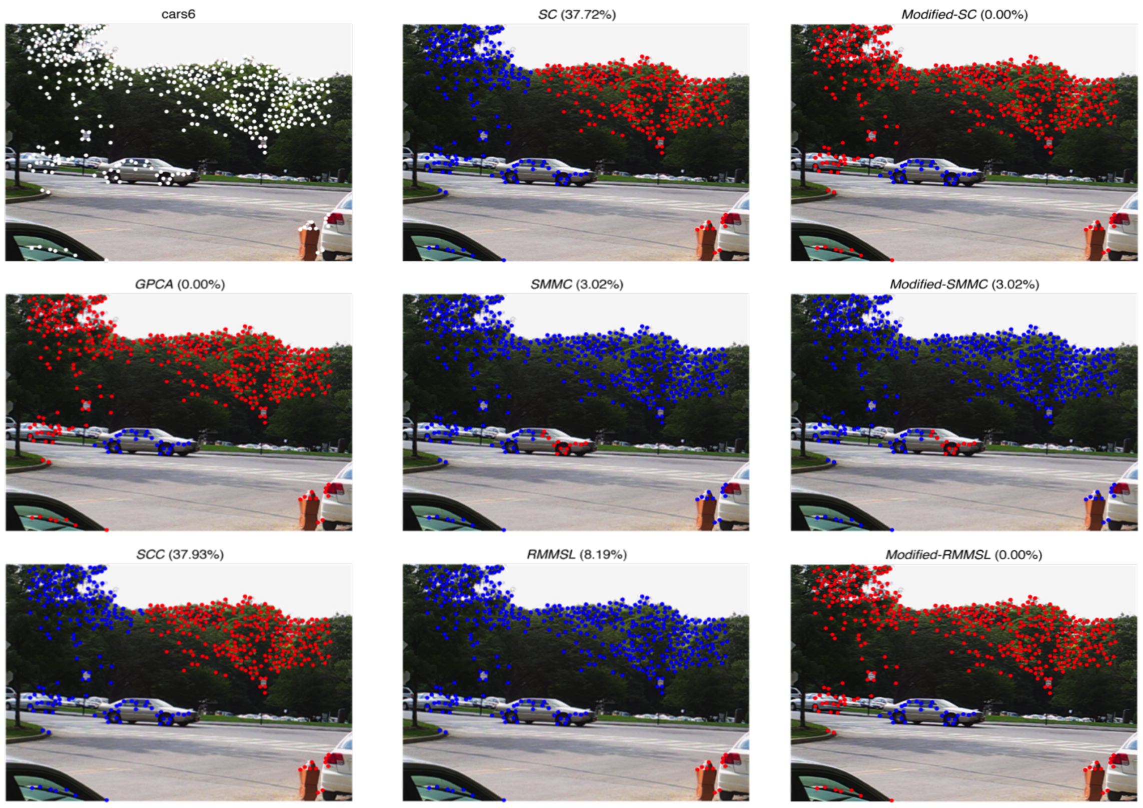


Fig. 3.8. Clustering results for cars6 video sequence. There are 31 frames in this data set. The car is crossing the frame and turning left at the same time.

sets, this is not necessarily the best way to choose parameters. In the cars_g13 data set, for example, it seems that several algorithms have poor performance. When we analysed this data set in detail, we found that the poor performance was related to the distribution of the points. The cars5_g13 data consists of 36 points for car one, and 307 points for the background. By visualising the data, we noticed that the background points are less localised than the car points, as shown in Fig. 3.10 (left). If we choose $K = 12$ there are in fact three clusters, as shown in Fig. 3.10 (right). Even though two of the three clusters are in the same subspace, they are not connected, thus explaining the poor performance of the algorithms. To address the problem we tried other parameters. For $K = 20$ we obtained better performance (13.11% and 0% for *SC* and *Modified-SC* respectively).

Although our modification generally improves the results over the standard algorithm, we noticed that in some cases the results are worse. Take the 1R2RCT_A data set in Table 3.1 as an example, the *Modified-SC* method has a higher misclassification rate than the original *SC* algorithm. After a detailed analysis, we noticed that for certain data points, the iterative Mahalanobis distance learning may introduce incorrect connections. This situa-

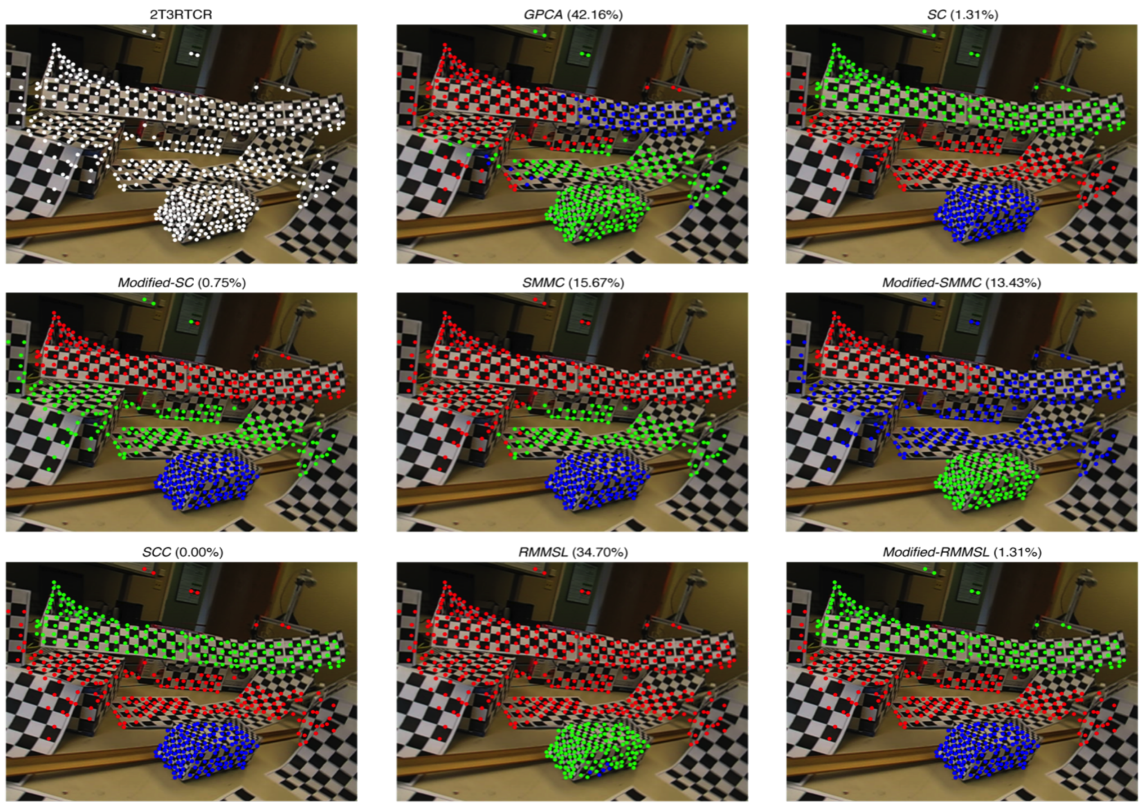


Fig. 3.9. Clustering results for 2T3RTCR video sequence. There are 26 frames in this data set. The camera is rotating in the video, and the long checkerboard in the middle is swaying.

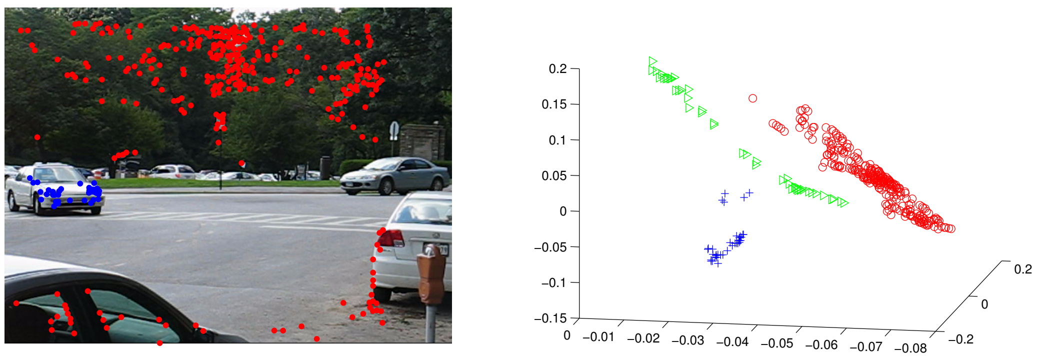


Fig. 3.10. The cars5_g13 data set. On the left we show the features provided with the data. On the right we show how the data is clustered into three groups in its projection into $\mathbb{R}^3$.

tion is shown in Fig. 3.11. The data shown in Fig. 3.11 has a complicated distribution, and this is probably the main reason that the modified neighbourhood selection method has failed. To some extent, this problem can be mitigated by allowing larger neighbourhoods. For example with $K = 20$ the misclassification rate is 3.56% for the *Modified-SC* algorithm and 4.04% for the *SC* algorithm. However, this is a minor improvement, and we tend to prefer a fixed choice of parameter for fair comparisons.

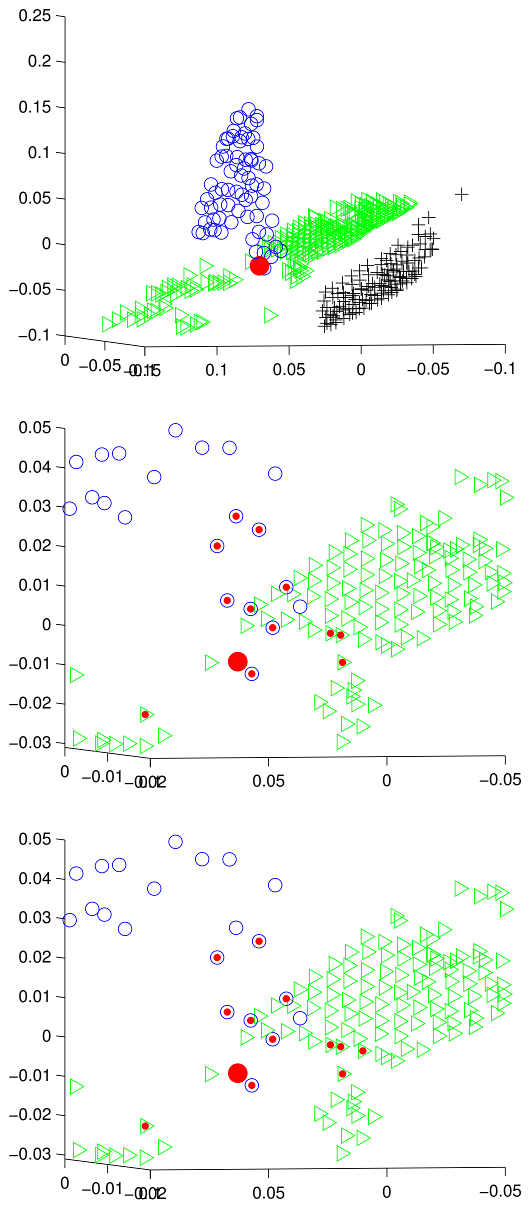


Fig. 3.11. Misconnections caused by Mahalanobis learning. On the top, we visualise the features in $\mathbb{R}^3$, with points belonging to different clusters distinguished by blue, green, and black. We want to find the neighbours of the red point which belongs to the blue cluster. In the middle, we show its neighbours, which have four misconnections, in red after one iteration. On the bottom, we show in red the neighbourhood learned by the iterative Mahalanobis algorithm. Notice it has introduced one more misconnection.

3.4.3 MNIST and COIL20 data sets

MNIST (LeCun, Bottou, Bengio, and Haffner, 1998) and COIL20 (Nene, Nayar, and Murase, 1996) are two other commonly used benchmark data sets for object recognition and clustering algorithms. MNIST is a data set of handwritten digits, with each digit represented by a 28×28 image matrix. The COIL20 data set has 20 objects, and each object has been captured by 72 images taken as the object is rotated by 5° increments. Some of the sample images are shown in Fig. 3.12. For these data sets, we compared the performance of the *SC* and *Modified-SC* algorithms since these two algorithms need the smallest number of parameters.

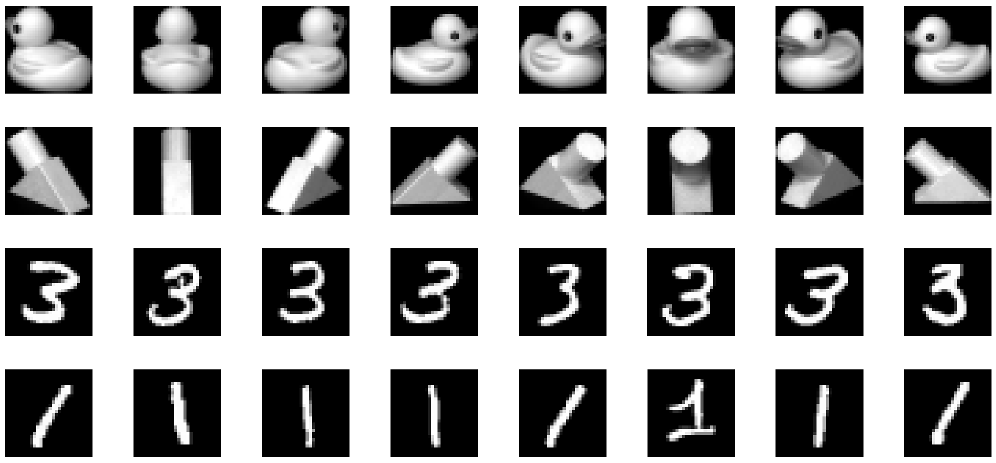


Fig. 3.12. Example pictures in COIL20 and MNIST data sets. The top two rows are example images taken from COIL20. The other images are taken from MNIST.

The misclassification rates are summarised in Fig. 3.13. The parameters D and K are tuned for each data set in order to achieve the best clustering performance. For COIL20, the whole data set is used for testing the performance of different clustering methods. We project the data points to $\mathbb{R}^D$, where D is 20, and the number of nearest neighbours K is 6. Since K-means algorithm is used in the final stage of the spectral clustering methods, we repeat the experiments 100 times and report the average misclassification rates. For MNIST, we randomly choose 50 data points from each digit. The parameters $D = 20$ and $K = 7$ are used for this data set. The experiment is repeated 100 times and the average results are presented. From Fig. 3.13, we can see that *Modified-SC* has lower misclassification rates for both COIL20 and MNIST data sets and the standard deviations show that the performance

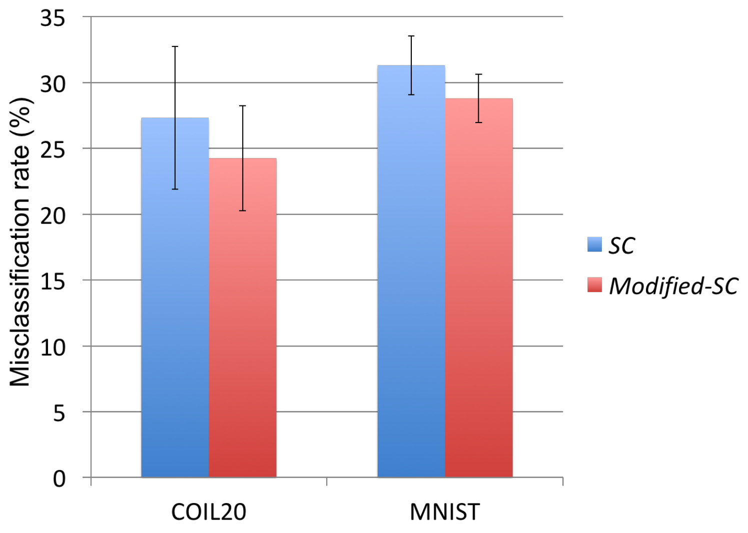


Fig. 3.13. Clustering performance in COIL20 and MNIST data sets. In COIL20, the misclassification rates for *SC* and Modified-SC are 27.33% and 24.25% respectively. The corresponding rates are 31.31% and 28.80% in MNIST. The error bars show the corresponding standard deviations of the misclassification rates.

of spectral clustering methods is relatively stable on the MNIST. In order to show the improvement is statistical significant, we adopt the two-sample t-test. The null hypothesis is that the misclassification rates from these two methods have the same means but their variances might be different. The alternative hypothesis is that the misclassification rates from *Modified-SC* has a smaller mean. The p-values for these two tests are 4.11×10^{-6} and 1.19×10^{-15} respectively. In practice, the null hypothesis is often rejected if the p-value is smaller than a significance level $\alpha_t = 0.05$. Thus, the alternative hypothesises are accepted in both tests and the neighbourhood identification method can be used to improve the quality of the similarity matrix W in spectral clustering method.

3.5 Summary

We have proposed an algorithm for using local information to learn an adaptive Mahalanobis distance, and thus provide high quality neighbourhoods for various algorithms. We have applied a modified neighbourhood selection method to spectral clustering, seeking improvement when the manifolds belonging to different clusters are near or intersect with each other. However, this type of neighbourhood selection could be used in any situation where higher quality edge connections are required.

We incorporated our proposed modification into some recent spectral clustering algorithms (Wang *et al.*, 2011; Gong *et al.*, 2012) and compared the performance of the modification against the original algorithms. In most cases we obtained improvements with real-world benchmark image data sets.

During our experiments, we noticed that in some cases the modification produces worse performance. We have analysed this phenomenon, noting that a change in parameters can often improve the situation.

Part II

Exploring geometric structures for manifold data

Chapter 4

Action Recognition based on Principal Geodesic Analysis

Note: Some portions of this chapter are taken from Fu *et al.* (2013).

In this chapter, we have investigated the use of geometric information for action recognition. Previously, Lui *et al.* (2010) proposed to represent each video sequence as a data point in some special manifold space and predict new video sequences by a nearest neighbour classifier. The computation required to label a query video is proportional to the size of the training data set, since we have to calculate the distances between the query video and all of the videos in the training data set. The high dimension of the manifold also makes the distance computation time consuming. This motivates us to explore geometric information for the recognition problem. Our approach for addressing this problem is to extract low dimensional subspace information for each class of the training videos, and then this information is used to label the query videos. Thus, the computation for labelling new video sequences depends on the number of classes rather than the size of the training data set.

4.1 Introduction

Action recognition in video sequences is an active research topic in computer vision. Its applications include surveillance system (Danafar and Gheissari, 2007), action based video annotation and retrieval (Poppe, 2010), and gesture-based human computer interfaces (Aggarwal and Ryoo, 2011). Fig. 4.1 presents some video sequences from the Cambridge Hand Gesture data set (Kim, Wong, and Cipolla, 2007), which consists of 9 gesture classes based on 3 different hand shapes (*Flat*, *Spread* and *V-shape*) and 3 different motions (*Leftward*, *Rightward* and *Contract*).

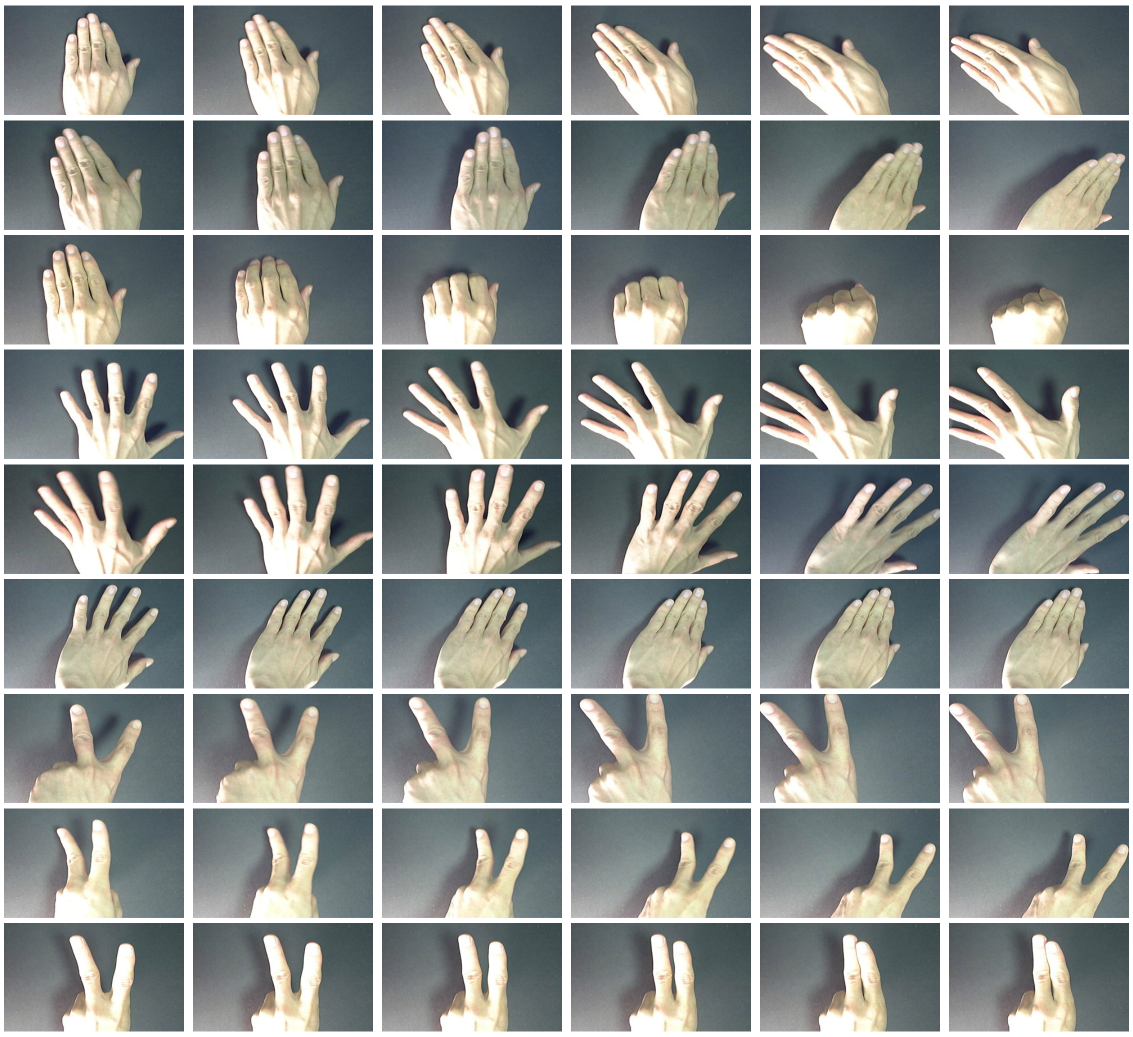


Fig. 4.1. Example pictures in the Cambridge Hand-Gesture data set. Each row shows some exemplar pictures from one video sequence. Here we show all of the 9 hand gestures: *Flat/Leftward*, *Flat/Rightward*, *Flat/Contract*, *Spread/Leftward*, *Spread/Rightward*, *Spread/Contract*, *V-shape/Leftward*, *V-shape/Rightward* and *V-shape/Contract* in each row respectively.

A key aspect of the problem is how to represent the video sequences and we briefly review three common methods: optical flow methods, interest-point methods and geometrical-structure methods. Efros, Berg, Mori, and Malik (2003) proposed a motion descriptor which is based on optical flow measurements of the video sequences, and then used a nearest neighbour algorithm for classification. Chaudhry *et al.* (2009) used the histogram of oriented optical flow (*HOOF*), which is scale and direction invariant, to represent each frame in the video sequences. Lin *et al.* (2009) proposed a shape-motion descriptor that is the concatenation of both optical flow information and shape information obtained either by silhouettes or the histogram of oriented gradients (*HOG*). Interest point methods include the work of Laptev and Lindeberg (2003) who generalised the Harris corner detector from 2D to 3D space, and proposed to extract space-time interest points (*STIP*) as 3D corners in the spatial-temporal video sequences.

The focus of this work is geometrical-structure methods which typically consider a video sequence as a point on a nonlinear manifold (Lui, 2012c). Kim *et al.* (2007) represented the video sequence as a 3D tensor, and introduced a canonical correlation analysis technique to measure the similarity between two video sequences. In two papers, Turaga *et al.* represented an action video as a point on a Grassmannian manifold and estimated a class-specific Gaussian distribution on this manifold from training examples (Turaga *et al.*, 2008, 2011). Classification is then made by assigning an example video to the most likely pre-learned class. Lui *et al.* (2010) proposed to view a video sequence as a point in a product space. They use the high order singular value decomposition (*HOSVD*) (De Lathauwer, De Moor, and Vandewalle, 2000) to factorise the tensor into a data point in the product of Grassmannian manifolds. In this way, the gesture recognition problem is changed to find the nearest point on the product space. In other papers, Lui *et al.* furthered the idea of action recognition in product spaces by using the tangent bundle structure and a nonlinear least squares regression framework respectively (Lui and Beveridge, 2011; Lui, 2012b).

In this work, noting that the dimension of Grassmannian manifolds is very high in Turaga's work (Turaga *et al.*, 2011) and in Lui's work (Lui, 2012d), we use principal geodesic analysis (*PGA*) (Fletcher, Lu, Pizer, and Joshi, 2004) on the Grassmannian manifold to reduce that dimensionality. In contrast to Turaga *et al.* (2011) who estimated the Gaussian distribution in high dimensions directly, our approach uses a low dimensional approximation of the data distribution. Lui (2012d) also exploited a low dimensional manifold structure, but it has a much higher computational complexity than our proposed method. We will compare our approach and the algorithm in Lui (2012d) in Section 4.3.

In Section 4.2, we introduce the necessary background for the proposed action recognition method, including details on *HOSVD*, computations on the Grassmannian manifold, the previous works on product space, and *PGA* on Grassmannian manifolds. In Section 4.3, we describe the proposed action recognition algorithm based on *PGA*. In Section 4.4, we test the proposed algorithm on several benchmark action data sets, and compare its performance with other action recognition algorithms. In Section 4.5, we conclude this chapter.

4.2 Background

4.2.1 High order singular value decomposition

The high order singular value decomposition (*HOSVD*) (De Lathauwer *et al.*, 2000) is a generalisation of singular value decomposition (*SVD*). For a 3D tensor T with size $n_1 \times n_2 \times n_3$, its *HOSVD* is represented as:

$$T = S \times_1 V^1 \times_2 V^2 \times_3 V^3 \tag{4.1}$$

where $S \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ is the core tensor, V^i is the orthogonal matrix which spans the row space of T unfolded along dimension i, and $\times_i$ is the tensor i-mode multiplication. The tensor unfolding operation converts a tensor into a matrix by using one dimension as the rows and stacking all other dimensions into the columns of the unfolded matrix. Fig. 4.2 shows an example of the unfolding along different dimensions. We use different colours to differentiate the unfolding in different directions. Note that the flattened tensor is a 2D matrix. The matrix V^i is obtained by the *SVD* of the i^{th} unfolding. The n-mode multiplication between a tensor $T \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_K}$ and a 2D matrix $M \in \mathbb{R}^{J_1 \times n_k}$ is defined by:

$$(T \times_n M)_{i_1 i_2 \cdots i_{n-1} j_1 i_{n+1} \cdots i_K} = \sum_{i_n=1}^{I_n} T_{i_1 i_2 \cdots i_{n-1} i_n i_{n+1} \cdots i_K} M_{j_1 i_n}. \tag{4.2}$$

From the definition of n-mode multiplication, we can see that it is actually the matrix multiplication along the n^{th} dimension of T by M. Finally, the core tensor is calculated by:

$$S = T \times_1 (V^1)' \times_2 (V^2)' \times_3 (V^3)'. \tag{4.3}$$

4.2.2 Distances on Grassmannian manifold

In Section 2.2.2, we have seen that the geodesic distance between two data points on a Grassmannian manifold can be calculated by Equation (2.13). Actually, this distance computation is based on the assumption that the Grassmannian manifold is equivalent to a quotient manifold ($R_*^{D\times d}/\sim$) and the canonical metric is used. In addition, there are alternative ways to view the Grassmannian manifold. For example, any data point $[X] \in \mathrm{Gr}(d, D)$ can be viewed as a data point XX' in $\mathcal{R}^{\mathcal{D}\times\mathcal{D}}$, i.e., viewing $\mathrm{Gr}(d, D)$ as a subset of $\mathcal{R}^{\mathcal{D}\times\mathcal{D}}$. With different embeddings as well as different metrics, we have different ways to define the distances between any two data points in the Grassmannian manifold.

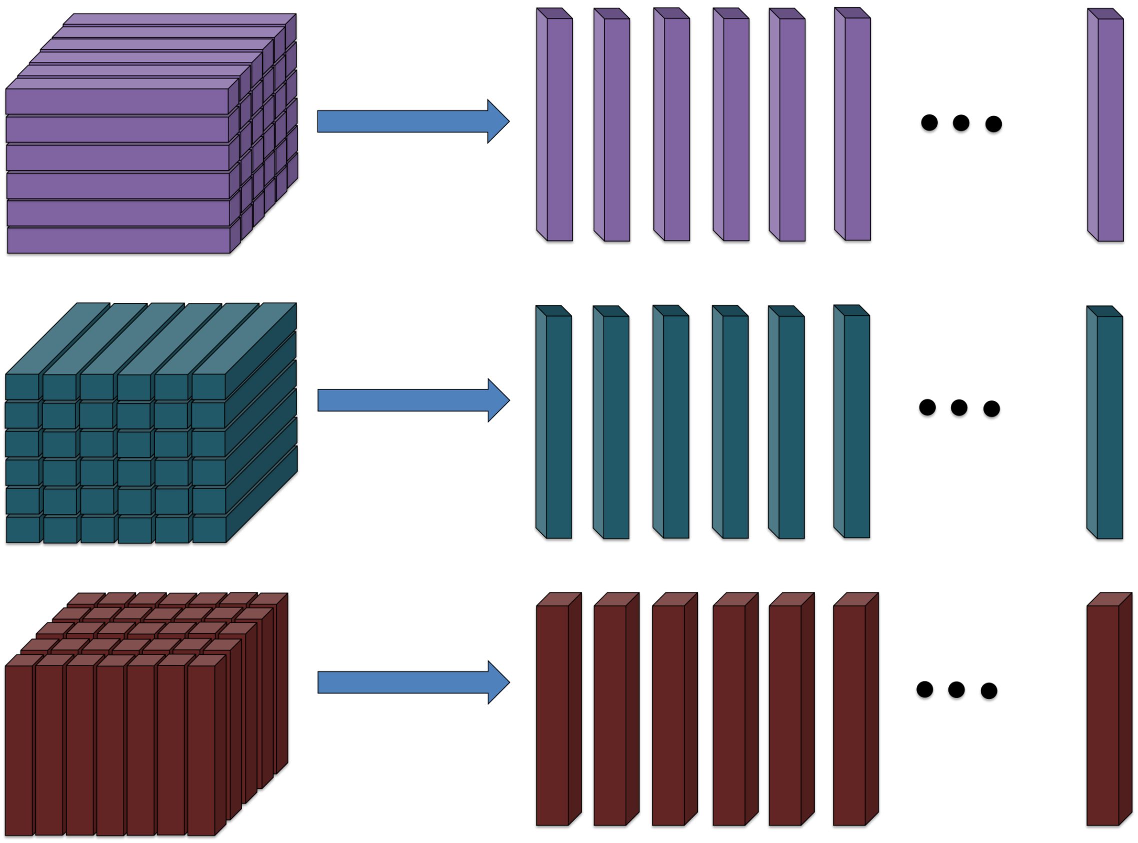


Fig. 4.2. Three ways to flatten a 3D tensor T. The tensor is partitioned into columns along the i^{th} dimension, and then the columns are aligned into a 2D matrix which is used to generate V^i in the *HOSVD* of T.

The calculation of different distances between two points on the Grassmannian manifold has an intimate relationship with the concept of principal angles which is a generalisation of angle information to high dimensions. Suppose S_1, S_2 are two subspaces. Let $r = \min\{\dim(S_1), \dim(S_2)\}$ be the smaller of these two dimensions, then the principal angles between S_1 and S_2, $\theta_1, \theta_2, \cdots, \theta_r$ satisfying $0 \leq \theta_1 \leq \theta_2 \leq \cdots \leq \theta_r$, are determined by the equations:

$$\begin{aligned} \cos(\theta_k) &= \max_{u_k \in S_1} \max_{v_k \in S_2} u'_k v_k \\ s.t.\ \ u'_k u_k &= v'_k v_k = 1,\ i = 1, 2, \cdots, k-1 \\ u'_k u_i &= v'_k v_i = 0,\ i = 1, 2, \cdots, k-1. \end{aligned} \tag{4.4}$$

Taking the chordal distance (Conway, Hardin, and Sloane, 1996; Turaga *et al.*, 2011) as an example, the distance between $[X]$ and $[Y]$ is defined as:

$$d([X], [Y]) = ||\sin\theta||_2 \tag{4.5}$$

where $\theta = [\theta_1, \theta_2, \cdots, \theta_r]$ is the vector of principal angles between the two linear subspaces spanned by the columns of X, Y respectively, and $\sin$ is applied component-wise.

4.2.3 Gesture recognition based on a product space

A video sequence can be naturally represented as a 3D tensor A. Each frame is viewed as a 2D matrix, and frames are stacked into a 3D matrix (Fig. 4.3). Lui *et al.* (2010) proposed that the 3D tensor representation of an action video can be viewed as a point in a product manifold (a product of Grassmannian manifolds) by taking *HOSVD* of this tensor.

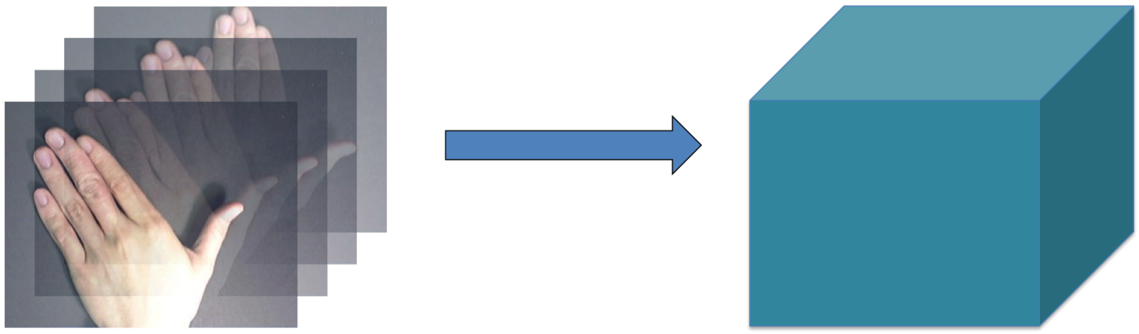


Fig. 4.3. Representing a video sequence as a 3D Tensor. Each frame is converted into a 2D grayscale matrix, and the frames are stacked into a 3D matrix.

For a given video tensor, A, its *HOSVD* is

$$A = S \times_1 V^1 \times_2 V^2 \times_3 V^3. \tag{4.6}$$

From the definition of *HOSVD*, V_i can be viewed as a data point in Grassmannian manifold Gr_i. Thus, the tensor A can be viewed as a point in the product space of $\mathrm{Gr}_1 \times \mathrm{Gr}_2 \times \mathrm{Gr}_3$. As an example, suppose a video sequence is represented as a 3D tensor A with size $20 \times 20 \times 32$. For a 3D tensor there are 3 possible unfolding, hence V^1, V^2 have dimension 20×640, and V^3 has dimension 32×400 in this case. This video sequence can be viewed as a point in the product space of $\mathrm{Gr}_1(20, 640) \times \mathrm{Gr}_2(20, 640) \times \mathrm{Gr}_3(32, 400)$.

Fig. 4.4 and Fig. 4.5 show two examples of such representations in the manifold space. The two video sequences are from the Cambridge Hand Gesture data set (Kim *et al.*, 2007). For each video sequence, each frame is resized to 64×64 pixels and 32 frames are used. In order to have a better visualisation, we approximate the *HOSVD* of A by rank 3, i.e., the three columns, which correspond to the largest eigenvalues, are used in V^i $(i = 1, 2, 3)$.

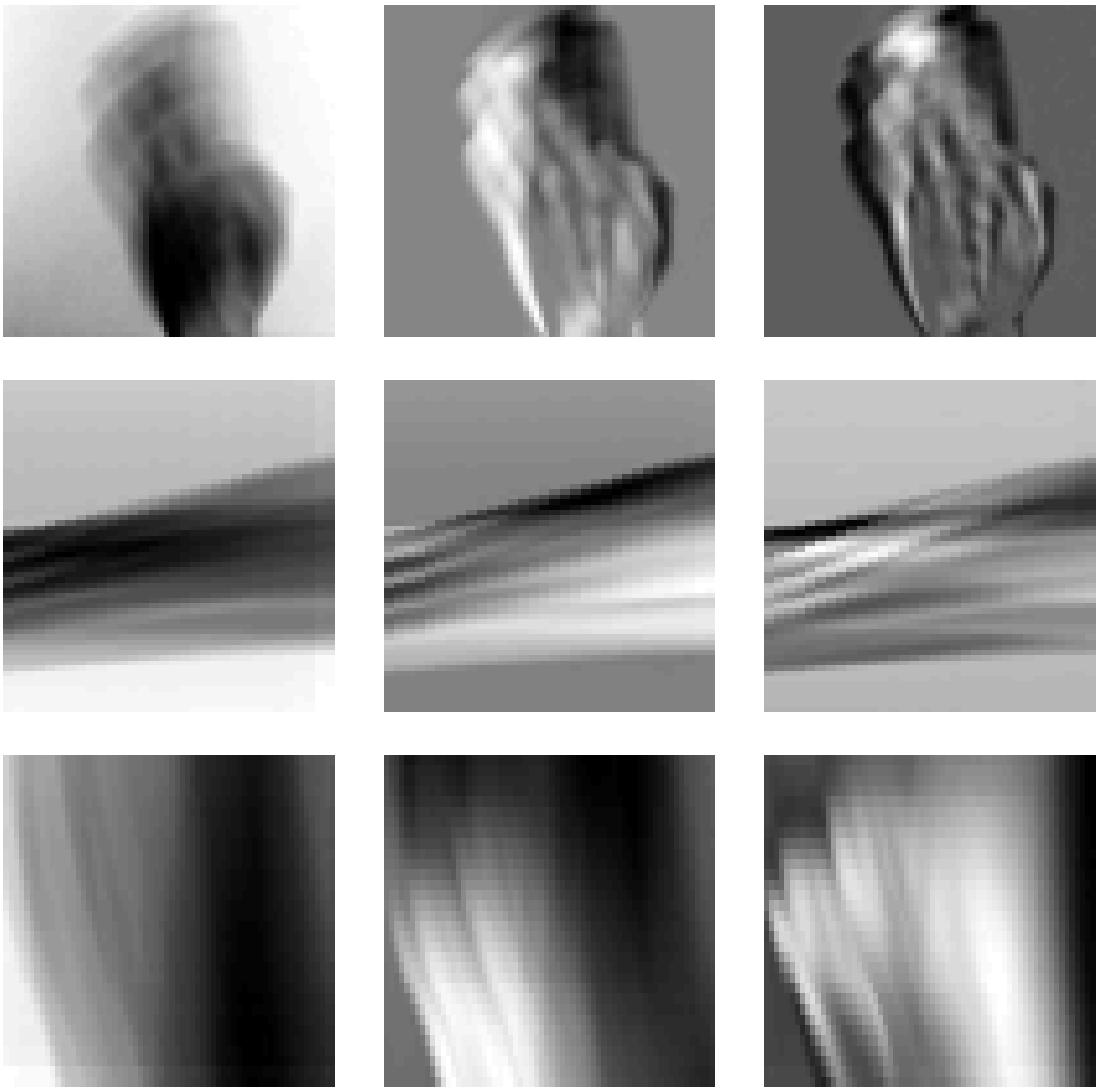


Fig. 4.4. Visualisation of a *Flat-Leftward* video sequence in the manifold space $\mathrm{Gr}_1 \times \mathrm{Gr}_2 \times \mathrm{Gr}_3$. Each row shows one data point in the corresponding Grassmannian manifold Gr_i. Here the subspace is represented by one of its basis sets, i.e., three mutual unit orthogonal vectors, and then each vector is reshaped into 2D pictures.

Each row in the figure corresponds to the three columns from the same V^i. Since these three columns are a set of unit basis for the subspace spanned by unfolding a video tensor, the i^{th} row corresponds to one data point in the Grassmannian manifold Gr_i. Besides, we can see that the first row is consistent with the common sense of the motion in a video sequence since it corresponds to the three components of the *PCA* decomposition of the video sequence. Because the video is not flattened along the time axis, the second and third row do not show the direct connection with the motion information, but, from the comparison between Fig. 4.4 and Fig. 4.5, we can see that the pattern for different motions is

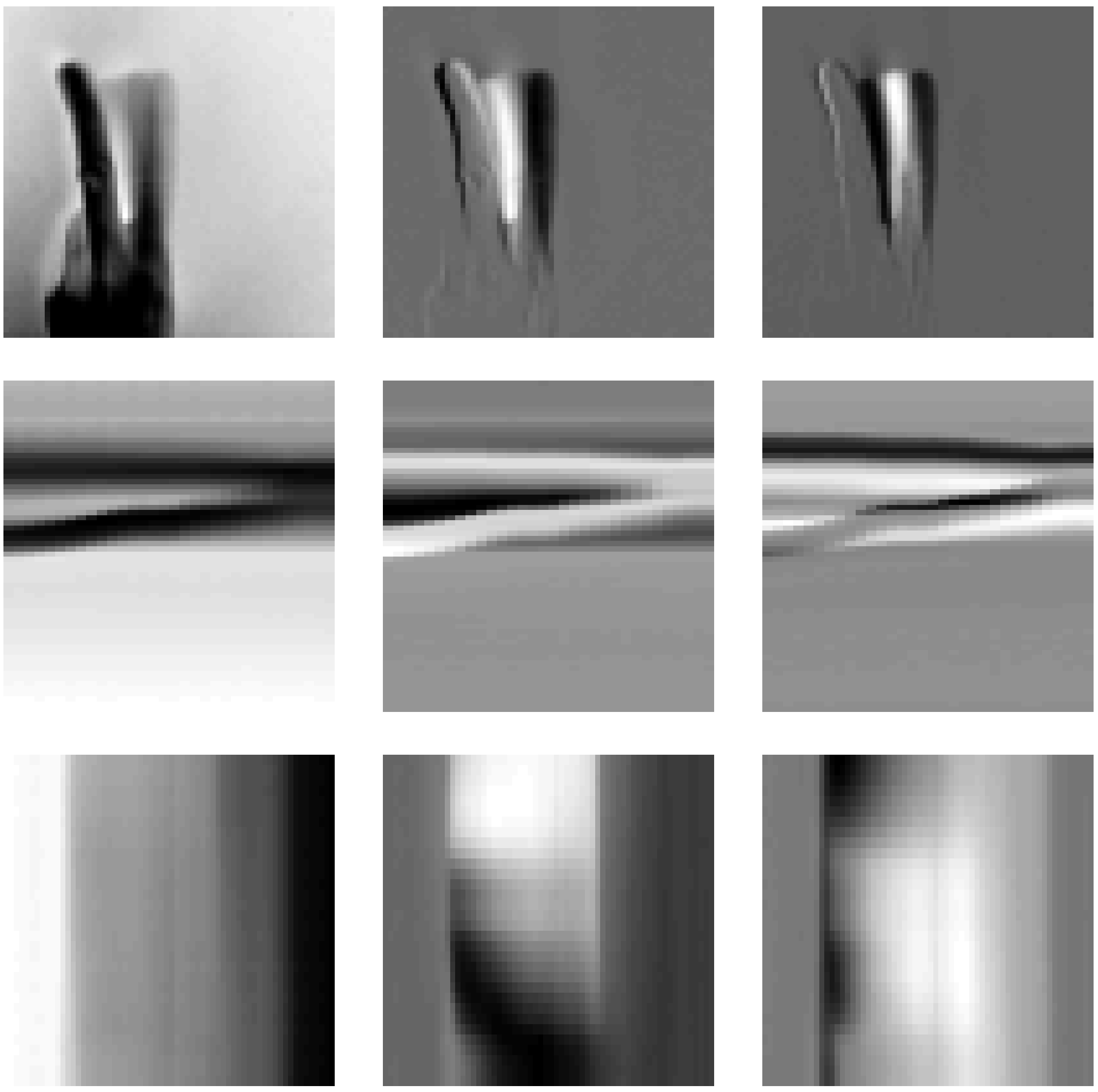


Fig. 4.5. Visualisation of a *V-shape-Contract* video sequence in the manifold space $\mathrm{Gr}_1 \times \mathrm{Gr}_2 \times \mathrm{Gr}_3$. Each row shows one data point in the corresponding Grassmannian manifold Gr_i. Here the subspace is represented by one of its basis sets, i.e., three mutual unit orthogonal vectors, and then each vector is reshaped into 2D pictures.

different. This means that the motion information is encoded into the subspace or the data point in Gr_i. Thus, each data point in the Gr_i captures an aspect of the motion information of the video sequence.

Lui *et al.* (2010) proposed a novel action recognition algorithm by using the geometric structure of the video sequences. They use the chordal distance to calculate the distances between the query video and video sequences in the training data set. The query video is predicted by the nearest neighbour classifier, i.e., it is predicted to have the same label as

Algorithm 3 Action Classification on Product Manifolds (Lui *et al.*, 2010)

```
    Input: $N$ training videos $A_1, A_2, \cdots, A_N$, one query video $A_{query}$
    Output: Label information for the query video $A_{query}$
1:  For each training video $A_i$, use the HOSVD technique, to compute $V_i^1, V_i^2, V_i^3$.
2:  Represent the query video $A_{query}$ as $V_{query}^1 \times V_{query}^2 \times V_{query}^3$.
3:  The distance between $A_i$ and $A_{query}$ is
        $d(A_i, A_{query}) = \sqrt{\sum_{j=1}^{3} d_g^2(V_i^j, V_{query}^j)}$
    where $d_g(\cdot,\cdot)$ is the chordal distance on the Grassmannian manifold (Equation (4.5)).
4:  Assign the label of video $A_{query}$ using NN-Classifier, i.e., the query video has the same
    label as its nearest neighbour video.
```

its nearest neighbour point. The whole process is summarised in Algorithm 3

Later, Lui *et al.* furthered the idea of the algorithm by incorporating the tangent bundle structure, i.e., Equation (2.13) is used for calculating distances between the query video and each video sequence in the training data set, of the Grassmannian manifold (Lui and Beveridge, 2011; Lui, 2012b) and introducing the least squares regression framework on the Grassmannian manifold (Lui, 2012a,d). The product space based recognition algorithms have several advantages, for example, they only use the raw pixels of the video sequences, and they have high recognition accuracy by utilising the geometrical structure of the Grassmannian manifold. On the other hand, as discussed in O'Hara (2013), the product space based algorithms have scalability limitations since these algorithms either have to calculate the distances between all of the training examples and the query video or have to do many iterations in each test. Therefore, they have a high computational cost when the number of training examples is large.

4.2.4 Principal geodesic analysis on a manifold space

Principal geodesic analysis (*PGA*) on nonlinear manifolds has been discussed in Fletcher *et al.* (2004). The main idea of *PGA* is to generalise the concepts of principal component analysis (*PCA*) to nonlinear manifolds. The general framework for *PGA* is formulated in Algorithm 4 where the operations $\exp$ and $\log$ are the exponential and logarithmic maps respectively. The key point of the generalisation of *PCA* to nonlinear manifolds is to make use of the tangent space by computing a point $\bar{x}$ on the manifold $\mathcal{M}$ which represents the mean of the data (in some sense) and then projecting all of the data points onto the tangent space $T_{\bar{x}}\mathcal{M}$. The mean of a data set in a manifold space is defined to minimise the optimisation objective:

$$C(x) = \frac{1}{2N}\sum_{i=1}^{N} d(x_i, x)^2. \tag{4.7}$$

Denote $\bar{x}$ as the current estimation of the mean. The gradients of $C(x)$ at $\bar{x}$ is the average of the data points when they are mapped to the tangent space $T_{\bar{x}}\mathcal{M}$ by the logarithmic map (Karcher, 1977), i.e.,

$$\nabla C(\bar{x}) = -\frac{1}{N}\sum_{i=1}^{N} \log_{\bar{x}} x_i. \tag{4.8}$$

Thus, the mean $\bar{x}$ is updated with $\exp_{\bar{x}}(\frac{\tau}{N}\sum_{i=1}^{N}\log_{\bar{x}} x_i)$ where τ is the step length of the optimisation. For Grassmannian manifold, the logarithmic map and the exponential map are calculated by Equation (2.12) and Equation (2.11) respectively. As discussed in Fletcher *et al.* (2004), we fix the $\tau = 1$ in our experiments. After calculating the covariance of the projected points on this tangent space, we follow the usual *PCA* procedure by calculating the corresponding eigenvectors and eigenvalues.

Algorithm 4 Principal Geodesic Analysis (Fletcher *et al.*, 2004)

```
    Input: $N$ data points $x_1, x_2, \cdots, x_N \in \mathbb{R}^n$
    Output: Principal directions $v_1, v_2, \cdots, v_k \in T_{\bar{x}}\mathcal{M}$, where $\bar{x}$ is the mean of the data points on the same manifold.
1:  initialize the mean $\bar{x} = x_1$ of the data points
2:  do
3:      $\nabla C(\bar{x}) = -\frac{1}{N}\sum_{i=1}^{N}\log_{\bar{x}} x_i$
4:      $\bar{x} \leftarrow \exp_{\bar{x}}(-\tau\nabla C(\bar{x}))$
5:  while $||\nabla C(\bar{x})|| > \varepsilon$
6:  Project the data points into the tangent space of $T_{\bar{x}}\mathcal{M}$ by $y_i = \log_{\bar{x}} x_i$, $i = 1, 2, \cdots, N$
7:  Calculate the covariance matrix $\mathrm{Cov} = \frac{1}{N}\sum_{i=1}^{N} y_i y_i'$
8:  Calculate the eigenvectors and eigenvalues of Cov.
9:  Return eigenvectors $v_1, v_2, \cdots, v_k$ which correspond to the leading $k$ eigenvalues.
```

4.3 Algorithm

4.3.1 Motivation

The Grassmannian manifolds, which are used to represent human actions in Liu's method, tend to have a very high dimension. The dimension of the Grassmannian manifold $\mathrm{Gr}(k, n)$ is $(n-k)k$. As an example, the dimension of the manifold $\mathrm{Gr}(32, 400)$ is $(400-32) \times 32 = 11776$. With the assumption that real data points tend to have hidden low dimensional structure, in this chapter we try to exploit this low dimensional structure in nonlinear manifolds in order to improve scalability and produce an effective algorithm for video-based recognition algorithms.

We are interested in investigating whether the low dimensional information extracted by *PGA* can be used for labelling the video sequences. The first step of *PGA* is to find the mean of the data points in the manifold. In Fig. 4.6 and Fig. 4.7, we have visualised means of the two video classes from Cambridge Hand Gesture data set (Kim *et al.*, 2007). We use the *Flat-Leftward* and *V-shape-Contract* classes from Set 5, and each of them consists of 20 video sequences. For better visualisation, we use a three dimensional subspace to represent each component of the product manifold. The mean of each class is calculated iteratively (Line 2 – Line 5 in Algorithm 4).

Compared to Fig. 4.4 and Fig. 4.5, which show specific two video sequences, Fig. 4.6 and Fig. 4.7 present the information summarised from the same video classes. From the figures, we can see that the mean captures the motion information of the video class, and the means for different classes are distinct with each other. Due to the similarity between the mean and the original representation of the video sequence, we speculate that all of the data points between them have similar motion information, i.e., the low dimensional space contains the information of some specific action class. Thus the subspace information for each video class can be used to label the query video sequences.

4.3.2 Algorithm

A visualisation of our approach is shown in Fig. 4.8. The top picture shows the offline training stage. For each class, we use Algorithm 4 to compute the mean, covariance structure, and a low-dimensional approximation of the class. The bottom picture shows the online testing stage. We compare the distance of the query data point to the extracted subspace of each class. The whole process is summarised in Algorithm 5. The first task can be cal-

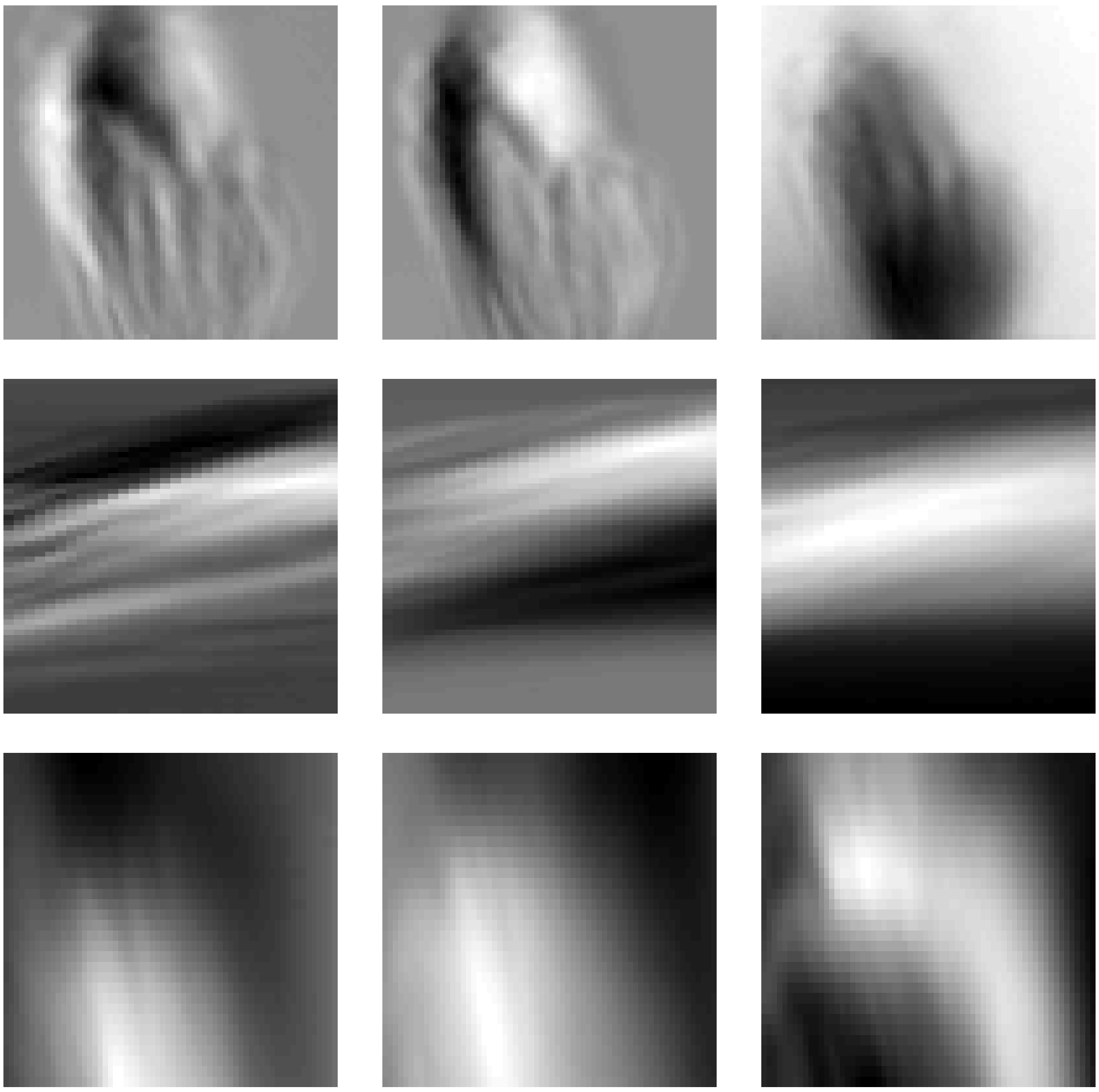


Fig. 4.6. Visualisation of the mean of the *Flat-Leftward* video class in the manifold space $\mathrm{Gr}_1 \times \mathrm{Gr}_2 \times \mathrm{Gr}_3$. We use 20 video sequences from Set 5 in Cambridge Hand Gesture data set. Each row shows the the data in the corresponding Grassmannian manifold Gr_i. Here the subspace is represented by one of its basis set, i.e., three mutual unit orthogonal vectors, and then each vector is reshaped into 2D pictures. Compared to the specific video sequence from the same video class, as it is shown in Fig. 4.4, the mean of the action class captures the motion information of this class since the pattern in corresponding rows is similar.

culated offline, and only needs to be run once for the training examples. The second task is computed online, and in the testing stage, we only need to compare the distance of the query data to the information extracted in the offline stage.

One of the advantages of this approach is that we can extract the classification information based on the training examples offline, and for testing the query videos, we classify

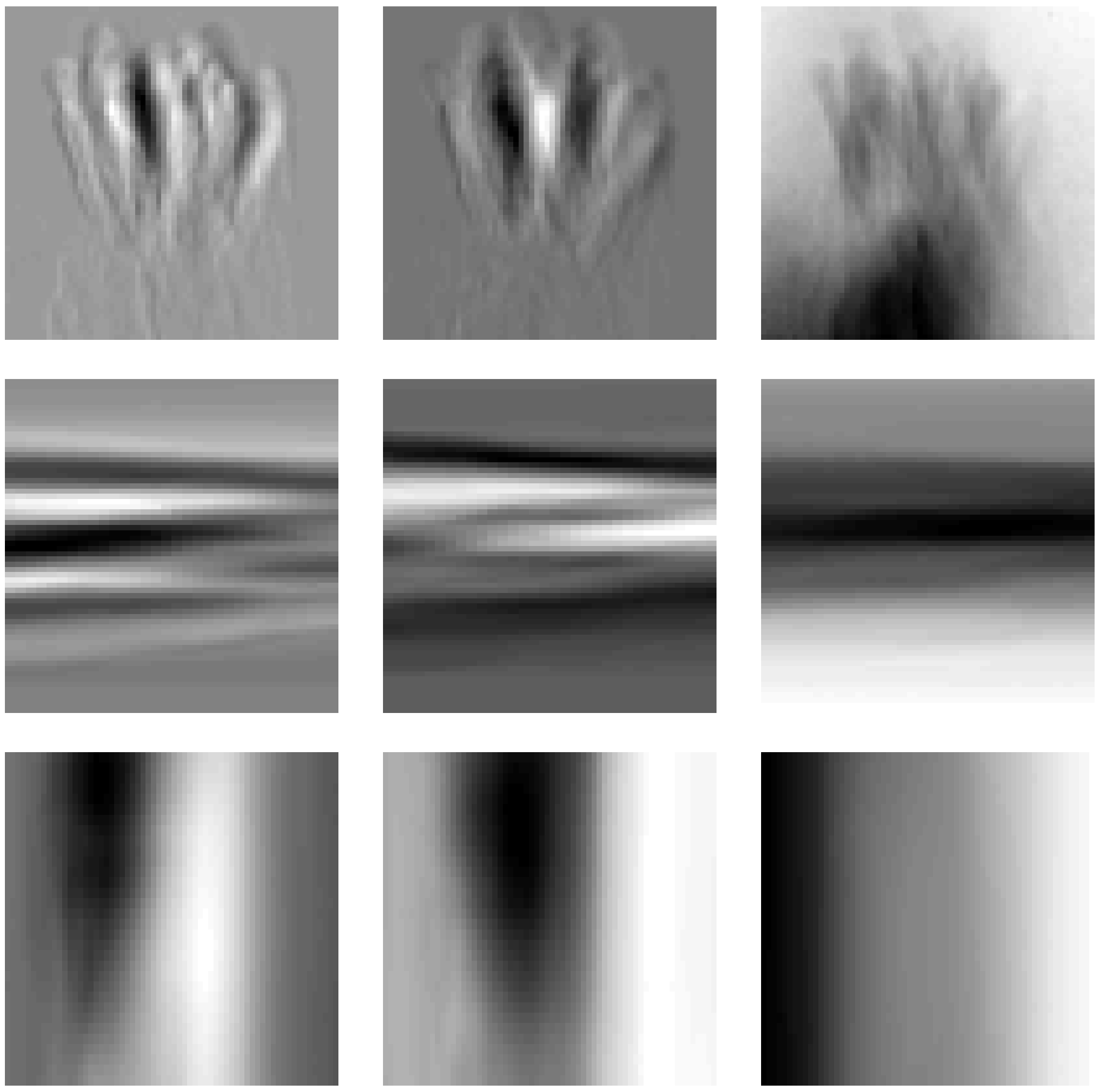


Fig. 4.7. Visualisation of the mean of the *V-shape-Contract* video class in the manifold space $\mathrm{Gr}_1 \times \mathrm{Gr}_2 \times \mathrm{Gr}_3$. We use 20 video sequences from Set 5 in Cambridge Hand Gesture data set. Each row shows the the data in the corresponding Grassmannian manifold Gr_i. Here the subspace is represented by one of its basis set, i.e., three mutual unit orthogonal vectors, and then each vector is reshaped into 2D pictures. Compared to the specific video sequence from the same video class, as it is shown in Fig. 4.5, the mean of the action class captures the motion information of this class since the pattern in corresponding rows is similar.

the query videos by using the extracted information rather than calculating the distance to each training example. Hence, the online calculating time is independent of the number of training examples, and linear in the number of classes and this makes the algorithm

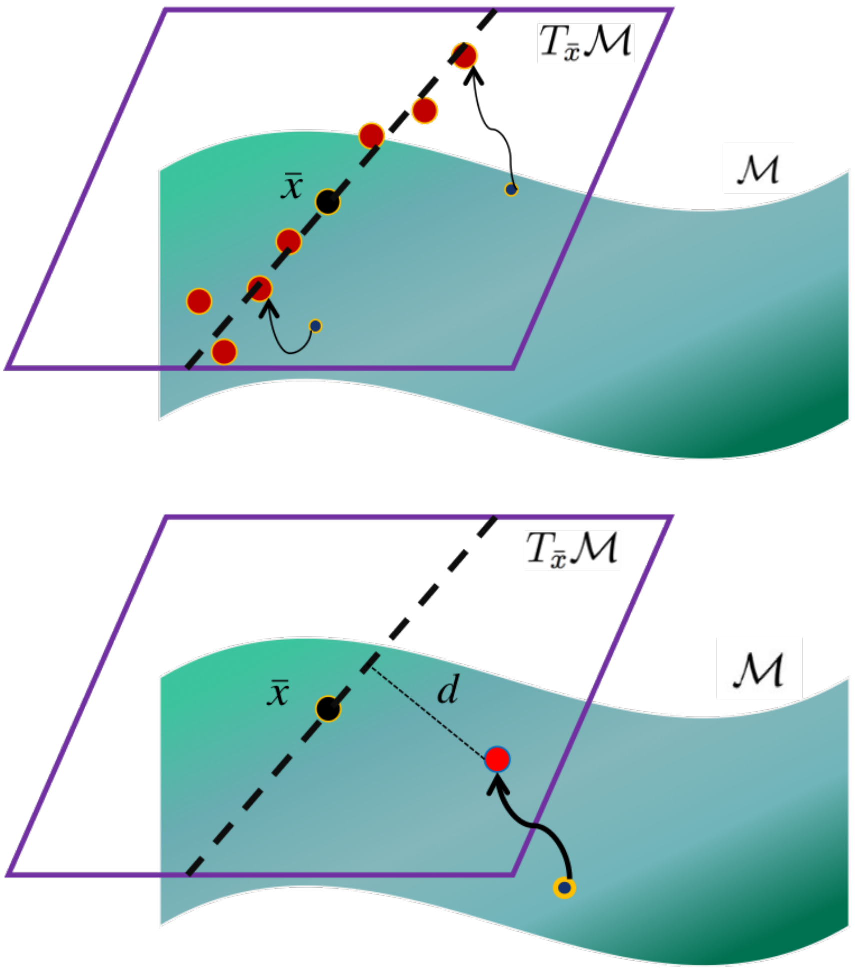


Fig. 4.8. Visualisation of the proposed method. The top picture displays the offline training stage. For each action class, the mean $\bar{x}$ of data points from this classed is located and then each data point is mapped to tangent space $T_{\bar{x}}\mathcal{M}$ by logarithmic map. The subspace of the data points in $T_{\bar{x}}\mathcal{M}$ is viewed as the geometric information of this action class. The bottom picture shows the online testing stage. For each action class, the query data point is mapped into the corresponding tangent space and its distance to the subspace extract in offline training stage is viewed as its distance to this action class.

scalable to large data sets.

The above algorithm utilises the appropriate tangent space structure, and then investigates the low dimensional subspace structure of the data distribution, i.e., we model a least squares regression on the tangent space. This idea shares some similarity with the work in (Lui, 2012d) and (Lui, 2012a), where non-linear least squares regression on a Grassmannian manifold was proposed. The main difference between these two approaches is that since the Grassmannian manifold is a non-linear space, the work in (Lui, 2012d) and (Lui, 2012a)

Algorithm 5 Action Classification based on Principal Geodesic Analysis

```
   Input: $N$ training videos $A_1, A_2, \cdots, A_N$, one query video $A_{query}$, suppose the label
   of the training video $A_i$ belongs to $\{1, 2, \cdots, C\}$.
   Output: Label information for the query video $A_{query}$.
1: For each class, $k$, use Algorithm 4 to compute the intrinsic means $\bar{A}_k^1, \bar{A}_k^2, \bar{A}_k^3$ and use
   PCA to find the low dimensional subspaces $S_k^1, S_k^2, S_k^3$ for the points of the class in the
   tangent spaces $T_{\bar{A}_k^1}, T_{\bar{A}_k^2}, T_{\bar{A}_k^3}$ of the three means respectively.
2: For the query video $A_{query}$, use HOSVD technique to compute $V_{query}^1, V_{query}^2, V_{query}^3$.
3: for $k = 1$ to $C$ do
4:     Project each $V_{query}^j, j \in \{1, 2, 3\}$, to the tangent space $T_{\bar{A}_k^j}$, and calculate its distance
       to the subspace $S_k^j$, denote the distance as $d^j(k, A_{query})$.
5: end for
6:
   $$\text{Label}_{A_{query}} = \underset{k=1\ldots C}{\arg\min} \sqrt{\sum_{j=1}^{3} \left(d^j(k, A_{query})\right)^2}$$
```

developed a kernel version of the least squares regression on the manifold directly. Our approach, by contrast, is based on a linear least squares regression by projecting the data to an appropriate tangent space. In addition, for classification of the query video sequence, the algorithm in (Lui, 2012d) and (Lui, 2012a) needs to do a weighted Karcher mean calculation, which is computationally expensive, for each query example. However, in our approach, the Karcher mean calculation is done offline, and we only need to test the query against each class of training examples. Thus the online computation of our approach is independent of the number of training examples.

In our experiments, we represent each video by a $20 \times 20 \times 32$ or $32 \times 32 \times 64$ tensor as commonly used in previous work (Lui, 2012b,d; O'Hara, 2013). This is mainly due to computational complexity considerations since the dimensionality of these representations are already 36,576 and 193,464 respectively.

As for the video representation step, take the tensor A with size $20 \times 20 \times 32$ as an example. The action region is located by manual cropping or human body detection techniques in each frame, and then the action region is resized to a 20×20 pixel size. Finally, we select the middle 32 frames of the video sequence or resize all of the frames to size 32, such as by interpolation techniques or sampling the frames uniformly. The steps of preparing a video tensor are visualised in Fig .4.9. We also have to decide the dimension of the linear subspace S_i in the tangent space. For the experiments we have done in Section 4.4, we set the dimension of S_i to be the same as the number of training examples in the corresponding class. This is because the number of training examples from each class in our experiments is much smaller than the dimension of the tangent space. In practice, when we have many training examples in each class, we suggest to set a threshold for the maximum dimension of the subspace.

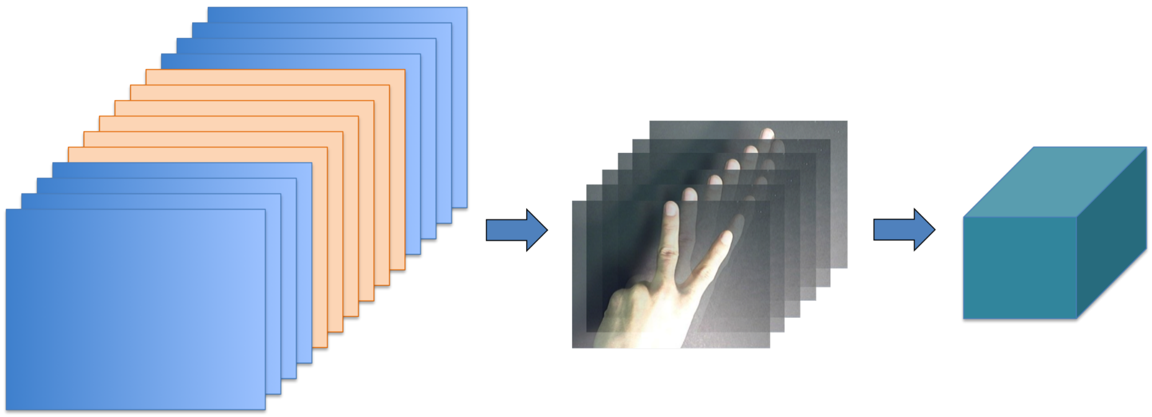


Fig. 4.9. Visualisation of preparing a video tensor A. The left picture shows a sequence of video frames. The first step is to locate the action regions. The frames, which are related to the gesture actions, are distinguished with yellow colour. The middle picture shows the extracted gesture action regions. Depending on different scenarios, the whole frame or only the action region is used. Finally, the 3D matrix, which consists of the pixel values, is converted to a video tensor A with fixed size.

For the implementation of the proposed approach, we need to calculate the exponential map and the logarithmic map multiple times. Since each video sequence is represented by the product of Grassmannian spaces, the exponential map and the logarithmic map in the product space can be calculated for each component and their product gives the final result (Lui, 2012b). Since each component of the product space is a Grassmannian manifold, the exponential map and the logarithmic map have analytic expressions (Equation (2.11) and Equation(2.12)).

For calculating the logarithm (Equation (2.12)), it is computationally expensive to calculate the orthogonal complement $X_\perp$ of $X \in \mathbb{R}^{D\times d}$, where D and d are the dimension of representative of the data points in Grassmannian manifold and they are fixed in the video representation step with the relationship $D \gg d$ in general, and X satisfies $X'X = I_d$. Here, we use the implementation suggested in Gallivan, Srivastava, Liu, and Dooren (2003). We find a rotation G for the column vectors of X, s.t. $XG = \begin{pmatrix} L \\ X_{02}G \end{pmatrix}$, where $L \in \mathbb{R}^{d\times d}$

is a lower triangular matrix with negative diagonal elements, and X is partitioned by $X = \begin{pmatrix} X_{01} \\ X_{02} \end{pmatrix}$. Then the orthogonal complement O of X is calculated by

$$O = I_n - \begin{pmatrix} L - I_d \\ X_{02}G \end{pmatrix} (I - L)^{-1}(L' - I_d \;\; G'X'_{02}). \tag{4.9}$$

In our implementation, the rotation G is obtained by calculating the QR decomposition of X_{01} which takes $\mathcal{O}(d^3)$ steps. Suppose $X_{01} = QR$, where $Q \in SO(d) = \{A \in \mathbb{R}^{d\times d} | A'A = I\}$ and R is an upper triangular matrix, we get G by flipping the columns of Q according to the sign of the diagonal element of R. The computation of the orthogonal complement O also takes $\mathcal{O}(d^3)$ time since we have to calculate the inverse of one matrix with size $d \times d$. Therefore the overall computation of calculating the orthogonal complement O is $\mathcal{O}(d^3)$.

4.4 Experiments

We compared the proposed method with previous action recognition algorithms (Lui *et al.*, 2010; Lui and Beveridge, 2011) on three action data sets: the Cambridge Hand Gesture data set (Kim *et al.*, 2007), the UMD Keck Body Gesture data set (Lin *et al.*, 2009) and the KTH human action data set (Schuldt, Laptev, and Caputo, 2004).

4.4.1 Cambridge Hand Gesture data set

There are 900 image sequences in Cambridge Hand Gesture data set (Kim *et al.*, 2007) and each gesture is displayed by 2 people under 5 different illumination conditions with 10 repetitions. This data set has a large intra-class variation in spatial and temporal alignment since each image sequence in this data set was recorded by a fixed camera with isolated gestures in space and time.

In this chapter, we follow the same data preprocessing steps as Lui *et al.* (2010), Lui (2012b) and Lui (2012d). For each image sequence, we choose the middle 32 pictures, and then resize each image into size 20×20. Therefore, each gesture video is represented by a 3D tensor with size $20 \times 20 \times 32$.

For this data set, we test the related algorithms with various numbers of training examples, and report the recognition accuracy (Fig. 4.10). The number of training examples for each class is up to 80. Since there are at most 20 video sequences for each class in each illumination condition, we use set 1 as testing data set, and the training examples are sampled from the following 4 sets. The accuracy results in Fig. 4.10 is averaged over 10 repetitions, and three methods use the same training examples in each repetition. When the size of the training set is small, the standard deviation of the recognition accuracy tends to be larger due to random sampling effects. When all of the video sequences are used, the standard deviation is 0 since the training data are always the same in each repetition. From this figure, we can see that the proposed approach has almost the same recognition accuracy as the algorithms proposed in Lui and Beveridge (2011) and Lui *et al.* (2010), and it has the highest recognition accuracy in 8 of 16 scenarios. When the size of the training examples is large enough, the algorithm in Lui and Beveridge (2011) has relatively better performance since the training examples in NN classifier can fully capture the action class's distribution.

In Fig. 4.11, we show the average computational time for each query. The algorithms are implemented in Matlab R2013b on an iMac with a 3.2GHz quad-core Intel Core i5 processor. In our implementation, the proposed method is the fastest in the testing stage when the number of training examples is large. Although the Matlab code is not optimised in our implementation, Fig. 4.11 shows that the proposed method has almost the same computation time regardless of the number of training examples. The average computation time of the algorithm in Lui and Beveridge (2011) and the algorithm in Lui *et al.* (2010) is proportional to the number of total training examples, whereas the average computation time of our proposed approach is proportional to the number of classes of the training examples.

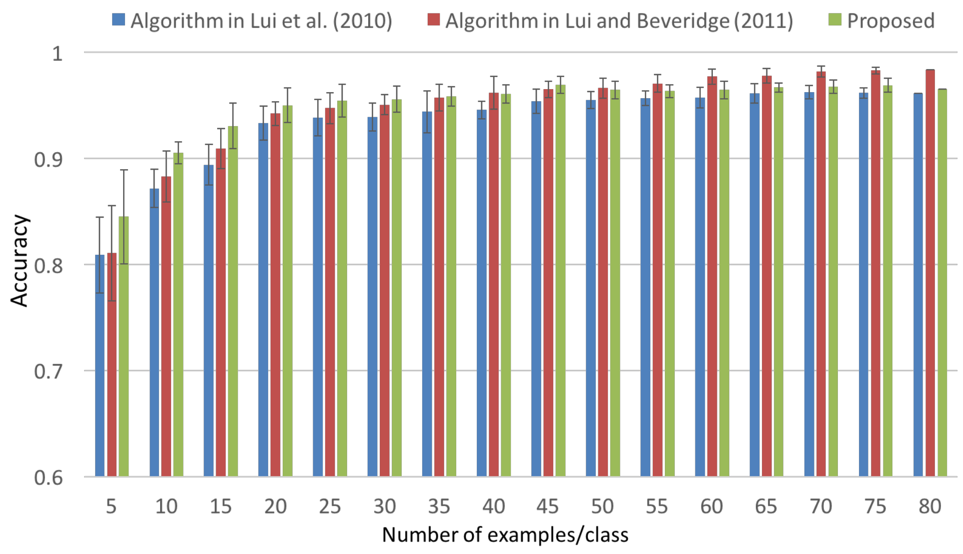


Fig. 4.10. Average recognition accuracy for labelling each query video with various numbers of training examples in each class. Each experiment repeats 10 times. The standard deviations are shown for the corresponding bars.

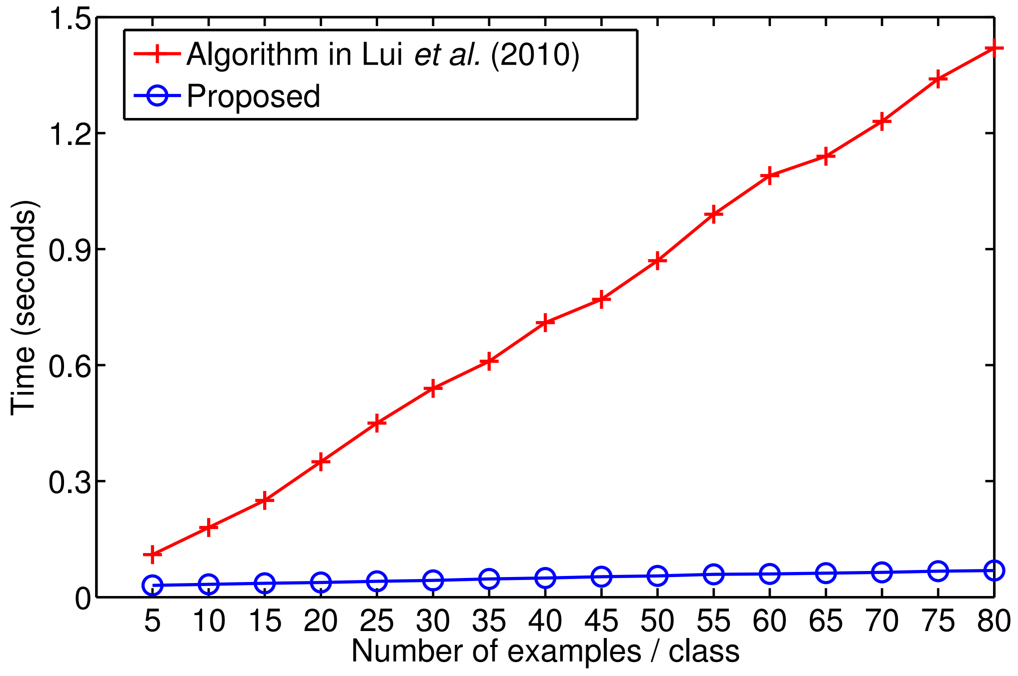


Fig. 4.11. Average computational time for labelling each query video with various numbers of training examples in each class. The x axis indicates the number of examples in each class, and the y axis shows the corresponding average computational time for each algorithm. We have not shown the average computational time of the algorithm in Lui and Beveridge (2011), which ranges from 1–15 seconds and also shows the linear relationship between the time and the number of training examples in each class.

4.4.2 UMD Keck Gesture data set

The UMD Keck Gesture data set (Lin *et al.*, 2009) is a collection of 14 different gestures, which are a subset of military signals, including the gestures for *Turn left*, *Turn right*, *Attention left*, *Attention right*, *Attention both*, *Stop left*, *Stop right*, *Stop both*, *Flap*, *Start*, *Go back*, *Close distance*, *Speed up* and *Come near*. These gestures are displayed by three people, and the person repeats the same gesture three times in each video sequence. The data set is recorded by a colour camera with 640×480 resolution. According to the background information, the data set is divided into two subsets: video sequences with static background and video sequences with dynamic background which are recorded by a moving camera or other objects moving around. In total, there are $3 \times 3 \times 14 = 126$ video sequences with static background, and $4 \times 3 \times 14 = 168$ sequences with dynamic background.

For the preprocessing of this data set, we have to segment the video sequences by each

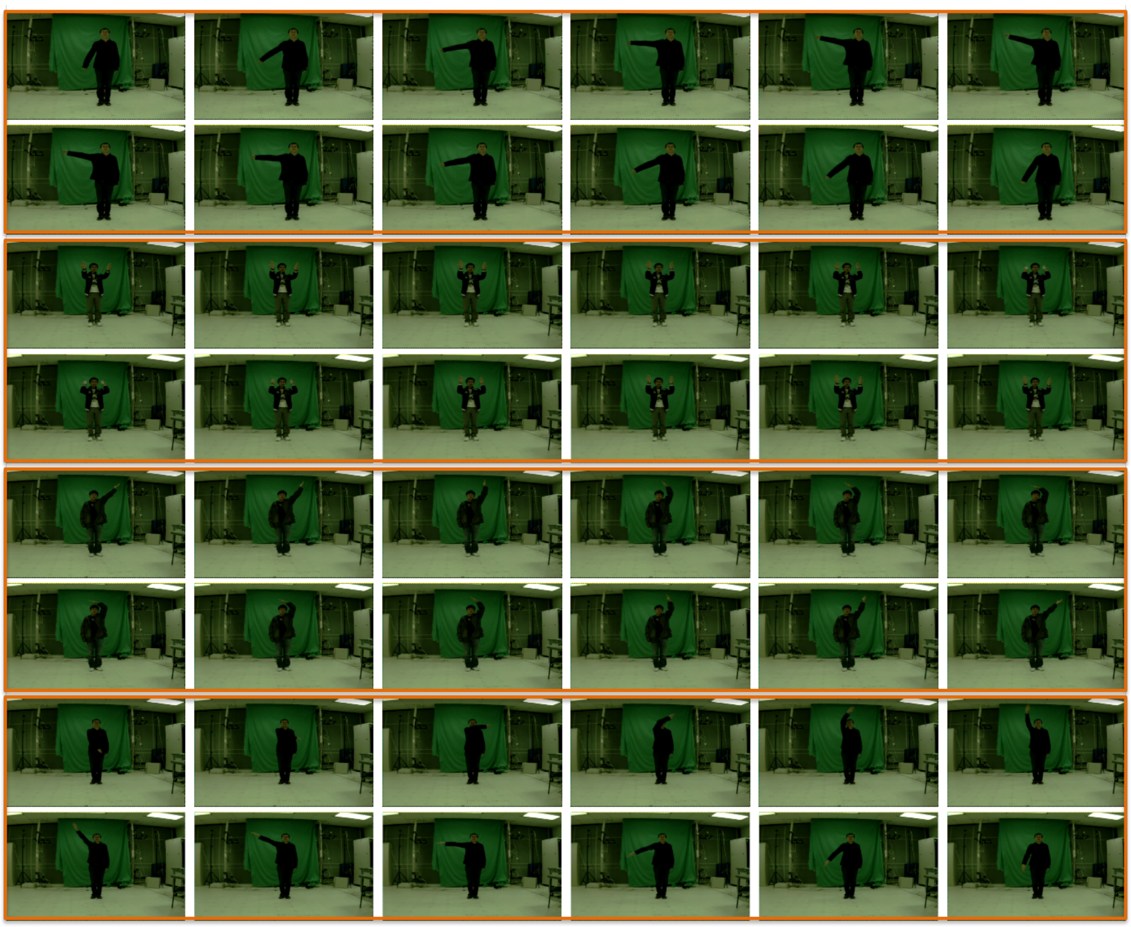


Fig. 4.12. Example video sequences in UMD Keck Gesture data set. Here we show 4 military signals: *Turn left*, *Come near*, *Attention right* and *Start*. Each of them occupies two rows respectively.

independent action, which has already been provided as a part of the data set, and localise the action region in each frame. Lui has shared with us the preprocessed data of the 126 sequences with static background. Each of these action sequences have square bounding boxes that we resized to size $32 \times 32 \times 64$. For the 168 sequences with dynamic background, we manually select the bounding box for the first frame in each video sequence, and the following position of the bounding boxes in each frame is tracked by the Real-Time Compressive Tracking algorithm (Zhang, Zhang, and Yang, 2012). After extracting the action region, we resize the 3D tensor into size $32 \times 32 \times 64$.

We test the performance of the proposed method according to the testing protocol in Jiang, Lin, and Davis (2012) and Lui (2012b). The first experiment is conducted on 126 video sequences with static background. We use the leave-one-out testing method where we select the gestures displayed by one person as testing examples, and use the remaining video sequences as training examples. The performance of different algorithms is shown in Table 4.1. In this data set, the proposed method has the highest recognition accuracy. The second experiment uses all of the videos with static background as training data points, and the video sequences with dynamic background are used as testing data. The recognition accuracy of the proposed algorithm achieves 93.4% which is again higher than the recognition accuracy of the other two algorithms (both are 92.8%). In both experiments, the proposed method takes similar amount of time (0.343 seconds and 0.350 seconds) to label each video sequence. The Algorithm in Lui *et al.* (2010) and Algorithm in Lui and Beveridge (2011), which are based on nearest neighbour classifier, take longer time to label the video sequences in the second experiment (0.607 to 0.9055 in Algorithm in Lui *et al.* (2010) and 18.014 to 27.068 in Algorithm in Lui and Beveridge (2011)).

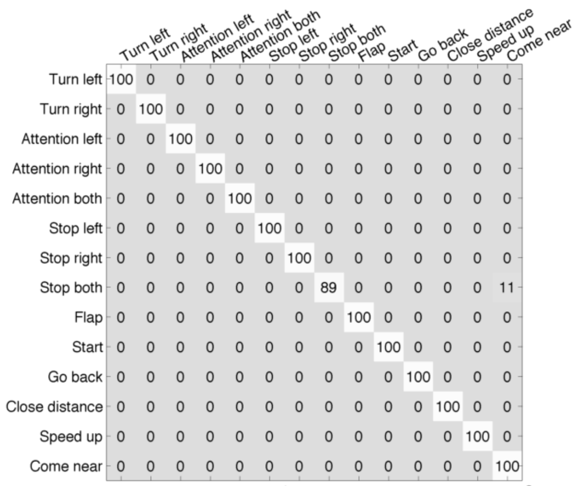
The confusion maps of the algorithms on both static and dynamic background are shown in Fig. 4.13. The confusion map details the prediction results including the information of mispredictions. From the maps, we can see that the *Stop both* class is relatively challenge across all of the algorithms. One interesting phenomenon is that although the accuracy on the *Stop both* class is 89% for these three algorithms, the pattern of misprediction is different. For example, the proposed method predicts all of the misclassified video sequences as *Come near*, whereas the other two method predict them to be *Stop left* and *Speed up* respectively. In addition to the improved overall accuracy of the proposed method, we also notice that the proposed method has the best performance for each action class.

	Person 1	Person 2	Person 3
Algorithm in Lui *et al.* (2010)	95.23%	97.61%	**100**$\%$
Algorithm in Lui and Beveridge (2011)	90.47%	97.61%	95.23%
Proposed	**97.61**$\%$	**100**$\%$	**100**$\%$

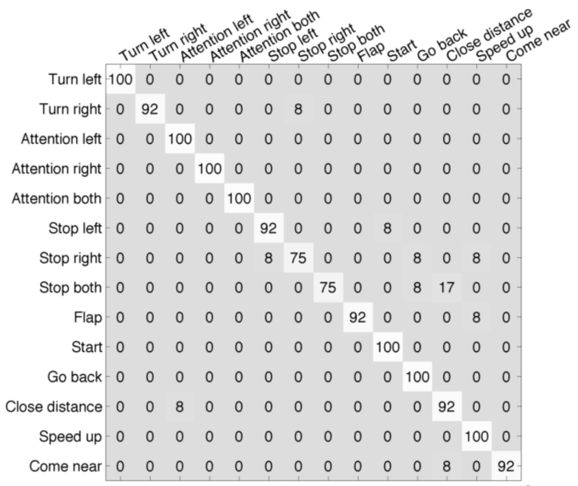
Tab. 4.1. Recognition results for video sequences with static background in Keck gesture data set.

4.4.3 KTH Human Action data set

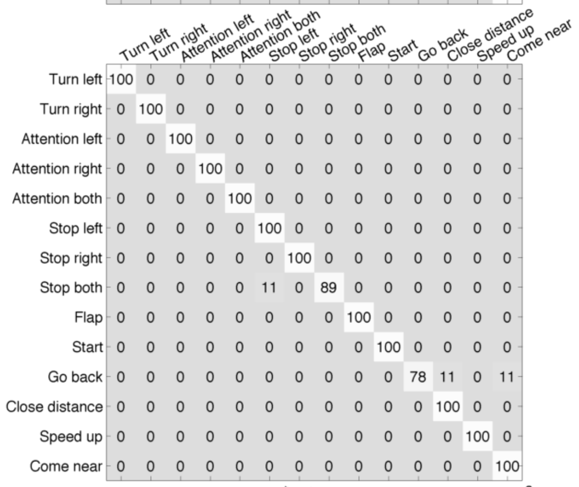
The KTH Human Action data set (Schuldt *et al.*, 2004) is a benchmark data set for human activity recognition. It has six different human activities including *Boxing*, *Hand clapping*, *Hand waving*, *Jogging*, *Running* and *Walking*. Each action is performed by 25 people in 4



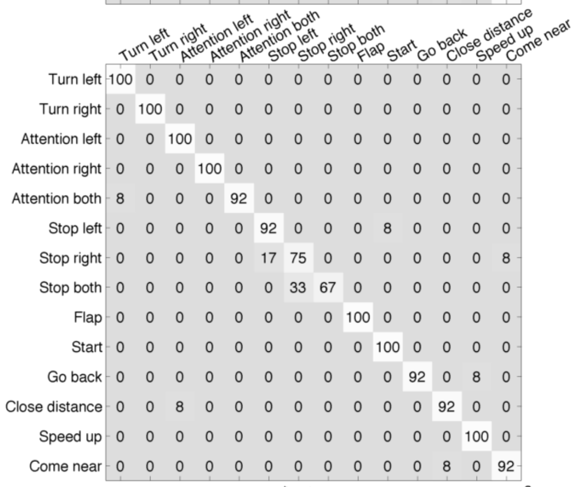
	Turn left	Turn right	Attention left	Attention right	Attention both	Stop left	Stop right	Stop both	Flap	Start	Go back	Close distance	Speed up	Come near
Turn left	100	0	0	0	0	0	0	0	0	0	0	0	0	0
Turn right	0	100	0	0	0	0	0	0	0	0	0	0	0	0
Attention left	0	0	100	0	0	0	0	0	0	0	0	0	0	0
Attention right	0	0	0	100	0	0	0	0	0	0	0	0	0	0
Attention both	0	0	0	0	100	0	0	0	0	0	0	0	0	0
Stop left	0	0	0	0	0	100	0	0	0	0	0	0	0	0
Stop right	0	0	0	0	0	0	100	0	0	0	0	0	0	0
Stop both	0	0	0	0	0	0	0	89	0	0	0	0	0	11
Flap	0	0	0	0	0	0	0	0	100	0	0	0	0	0
Start	0	0	0	0	0	0	0	0	0	100	0	0	0	0
Go back	0	0	0	0	0	0	0	0	0	0	100	0	0	0
Close distance	0	0	0	0	0	0	0	0	0	0	0	100	0	0
Speed up	0	0	0	0	0	0	0	0	0	0	0	0	100	0
Come near	0	0	0	0	0	0	0	0	0	0	0	0	0	100



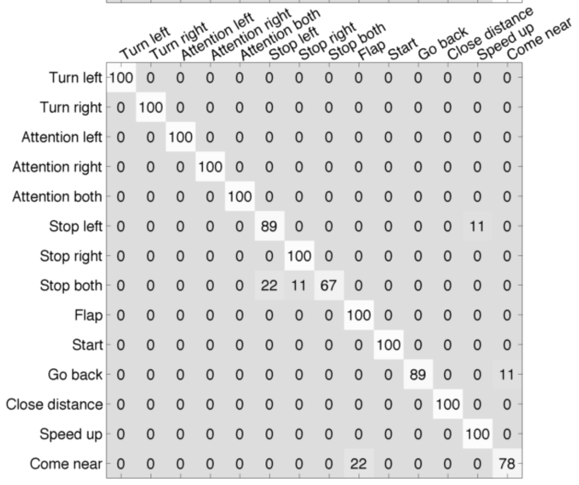
	Turn left	Turn right	Attention left	Attention right	Attention both	Stop left	Stop right	Stop both	Flap	Start	Go back	Close distance	Speed up	Come near
Turn left	100	0	0	0	0	0	0	0	0	0	0	0	0	0
Turn right	0	92	0	0	0	0	8	0	0	0	0	0	0	0
Attention left	0	0	100	0	0	0	0	0	0	0	0	0	0	0
Attention right	0	0	0	100	0	0	0	0	0	0	0	0	0	0
Attention both	0	0	0	0	100	0	0	0	0	0	0	0	0	0
Stop left	0	0	0	0	0	92	0	0	0	8	0	0	0	0
Stop right	0	0	0	0	0	8	75	0	0	0	8	0	8	0
Stop both	0	0	0	0	0	0	0	75	0	0	8	17	0	0
Flap	0	0	0	0	0	0	0	0	92	0	0	0	8	0
Start	0	0	0	0	0	0	0	0	0	100	0	0	0	0
Go back	0	0	0	0	0	0	0	0	0	0	100	0	0	0
Close distance	0	0	8	0	0	0	0	0	0	0	0	92	0	0
Speed up	0	0	0	0	0	0	0	0	0	0	0	0	100	0
Come near	0	0	0	0	0	0	0	0	0	0	0	8	0	92



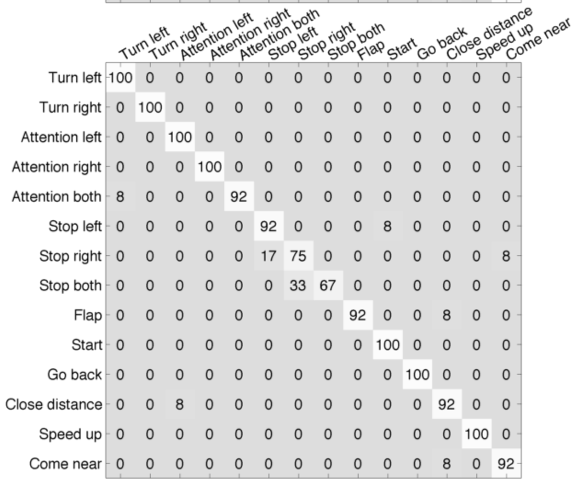
	Turn left	Turn right	Attention left	Attention right	Attention both	Stop left	Stop right	Stop both	Flap	Start	Go back	Close distance	Speed up	Come near
Turn left	100	0	0	0	0	0	0	0	0	0	0	0	0	0
Turn right	0	100	0	0	0	0	0	0	0	0	0	0	0	0
Attention left	0	0	100	0	0	0	0	0	0	0	0	0	0	0
Attention right	0	0	0	100	0	0	0	0	0	0	0	0	0	0
Attention both	0	0	0	0	100	0	0	0	0	0	0	0	0	0
Stop left	0	0	0	0	0	100	0	0	0	0	0	0	0	0
Stop right	0	0	0	0	0	0	100	0	0	0	0	0	0	0
Stop both	0	0	0	0	0	11	0	89	0	0	0	0	0	0
Flap	0	0	0	0	0	0	0	0	100	0	0	0	0	0
Start	0	0	0	0	0	0	0	0	0	100	0	0	0	0
Go back	0	0	0	0	0	0	0	0	0	0	78	11	0	11
Close distance	0	0	0	0	0	0	0	0	0	0	0	100	0	0
Speed up	0	0	0	0	0	0	0	0	0	0	0	0	100	0
Come near	0	0	0	0	0	0	0	0	0	0	0	0	0	100



	Turn left	Turn right	Attention left	Attention right	Attention both	Stop left	Stop right	Stop both	Flap	Start	Go back	Close distance	Speed up	Come near
Turn left	100	0	0	0	0	0	0	0	0	0	0	0	0	0
Turn right	0	100	0	0	0	0	0	0	0	0	0	0	0	0
Attention left	0	0	100	0	0	0	0	0	0	0	0	0	0	0
Attention right	0	0	0	100	0	0	0	0	0	0	0	0	0	0
Attention both	8	0	0	0	92	0	0	0	0	0	0	0	0	0
Stop left	0	0	0	0	0	92	0	0	0	8	0	0	0	0
Stop right	0	0	0	0	0	17	75	0	0	0	0	0	0	8
Stop both	0	0	0	0	0	0	33	67	0	0	0	0	0	0
Flap	0	0	0	0	0	0	0	0	100	0	0	0	0	0
Start	0	0	0	0	0	0	0	0	0	100	0	0	0	0
Go back	0	0	0	0	0	0	0	0	0	0	92	0	8	0
Close distance	0	0	8	0	0	0	0	0	0	0	0	92	0	0
Speed up	0	0	0	0	0	0	0	0	0	0	0	0	100	0
Come near	0	0	0	0	0	0	0	0	0	0	0	8	0	92



	Turn left	Turn right	Attention left	Attention right	Attention both	Stop left	Stop right	Stop both	Flap	Start	Go back	Close distance	Speed up	Come near
Turn left	100	0	0	0	0	0	0	0	0	0	0	0	0	0
Turn right	0	100	0	0	0	0	0	0	0	0	0	0	0	0
Attention left	0	0	100	0	0	0	0	0	0	0	0	0	0	0
Attention right	0	0	0	100	0	0	0	0	0	0	0	0	0	0
Attention both	0	0	0	0	100	0	0	0	0	0	0	0	0	0
Stop left	0	0	0	0	0	89	0	0	0	0	0	0	11	0
Stop right	0	0	0	0	0	0	100	0	0	0	0	0	0	0
Stop both	0	0	0	0	0	22	11	67	0	0	0	0	0	0
Flap	0	0	0	0	0	0	0	0	100	0	0	0	0	0
Start	0	0	0	0	0	0	0	0	0	100	0	0	0	0
Go back	0	0	0	0	0	0	0	0	0	0	89	0	0	11
Close distance	0	0	0	0	0	0	0	0	0	0	0	100	0	0
Speed up	0	0	0	0	0	0	0	0	0	0	0	0	100	0
Come near	0	0	0	0	0	0	0	0	22	0	0	0	0	78



	Turn left	Turn right	Attention left	Attention right	Attention both	Stop left	Stop right	Stop both	Flap	Start	Go back	Close distance	Speed up	Come near
Turn left	100	0	0	0	0	0	0	0	0	0	0	0	0	0
Turn right	0	100	0	0	0	0	0	0	0	0	0	0	0	0
Attention left	0	0	100	0	0	0	0	0	0	0	0	0	0	0
Attention right	0	0	0	100	0	0	0	0	0	0	0	0	0	0
Attention both	8	0	0	0	92	0	0	0	0	0	0	0	0	0
Stop left	0	0	0	0	0	92	0	0	0	8	0	0	0	0
Stop right	0	0	0	0	0	17	75	0	0	0	0	0	0	8
Stop both	0	0	0	0	0	0	33	67	0	0	0	0	0	0
Flap	0	0	0	0	0	0	0	0	92	0	0	8	0	0
Start	0	0	0	0	0	0	0	0	0	100	0	0	0	0
Go back	0	0	0	0	0	0	0	0	0	0	100	0	0	0
Close distance	0	0	8	0	0	0	0	0	0	0	0	92	0	0
Speed up	0	0	0	0	0	0	0	0	0	0	0	0	100	0
Come near	0	0	0	0	0	0	0	0	0	0	0	8	0	92

Fig. 4.13. Confusion matrix for Keck data set. The first column shows the recognition accuracy results for static background data set, and the second column shows the results for dynamic background data set. The first row is the classification accuracy results of proposed method. The second and the third rows are the results from Algorithm in Lui *et al.* (2010) and Algorithm in Lui and Beveridge (2011) respectively.

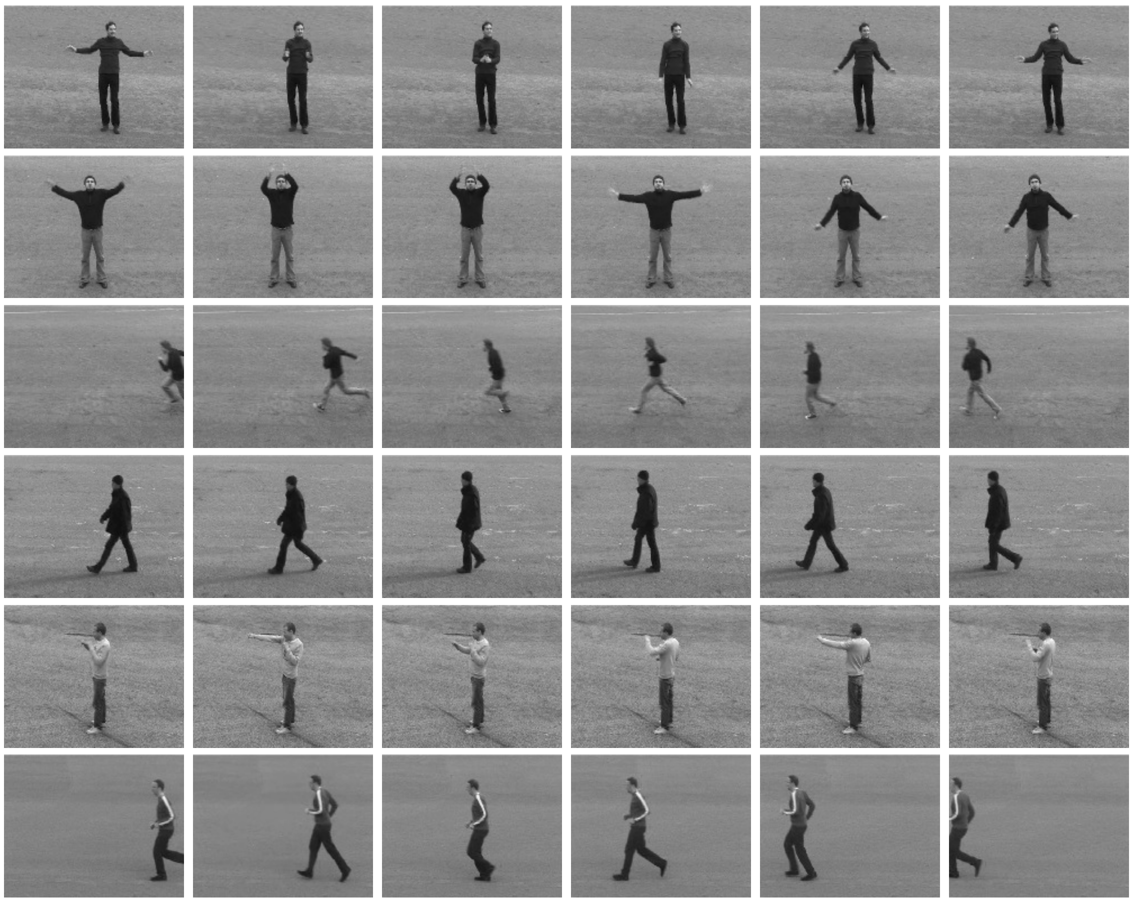


Fig. 4.14. Screenshots in the KTH Human Action data set. Here we show six different human actions: *Hand clapping*, *Hand waving*, *Running*, *Walking*, *Boxing*, and *Jogging* in each row correspondingly.

different scenarios, including outdoors (Scene 1), outdoors with scale variation (Scene 2), outdoors with different clothes (Scene 3) and indoors (Scene 4). Some example frames are shown in Fig. 4.14. We use 598 video sequences for training and testing (there are 600 videos in this data set, and we noticed that one video is missing and one other could not be opened in our downloaded data set).

The common protocol for the preprocessing of this data set is to segment the video sequences into independent human activities and then extract the action region. The action region extraction is often done by manually cropping or using human body detection techniques (Kim and Cipolla, 2009). Lui shared with us their video sequences segmentation information and the position of the bounding box. After extracting the action region, each action is resized into a 3D tensor with size $20 \times 20 \times 32$.

For this data set, we test the three algorithms using leave-one-out. Table 4.2 presents the recognition accuracy of the three algorithms. Our proposed approach has similar performance to the algorithms from Lui *et al.* (2010) and Lui and Beveridge (2011) except the second scenario. This is possibly due to the fact that the videos in scene 2 are taken with scale variation and we use the fixed space-time bounding box to capture the action region. Compared to our approach, the algorithms in Lui *et al.* (2010) and Lui and Beveridge (2011) classify the query video by calculating its distance to each training example which might lead to a higher recognition accuracy. The confusion maps of the proposed method on these four scenarios are shown in Fig. 4.15. From the confusion maps, we can see that the prediction accuracy on scenario 1 is highest across all of the six action classes. Although the accuracy is comparatively lower in scenarios 2, we notice that the prediction for the *Running* class is highest when compared to the results in other scenarios. This is possibly due to the fact that *Running* is relatively unaffected by scale variations. For other action classes, such as *Boxing*, *Hand clapping*, *Hand waving* and *Walking*, the scale variation deteriorates the recognition accuracy due to the different distributions of the training and testing data sets. Besides, the confusion matrixes in Fig. 4.15 also show the relationship between different action classes. For example, across the four scenes, video sequences from *Boxing* class are often classified as *Hand clapping* class, while video sequences from *Hand clapping* class are often classified as *Hand waving* class.

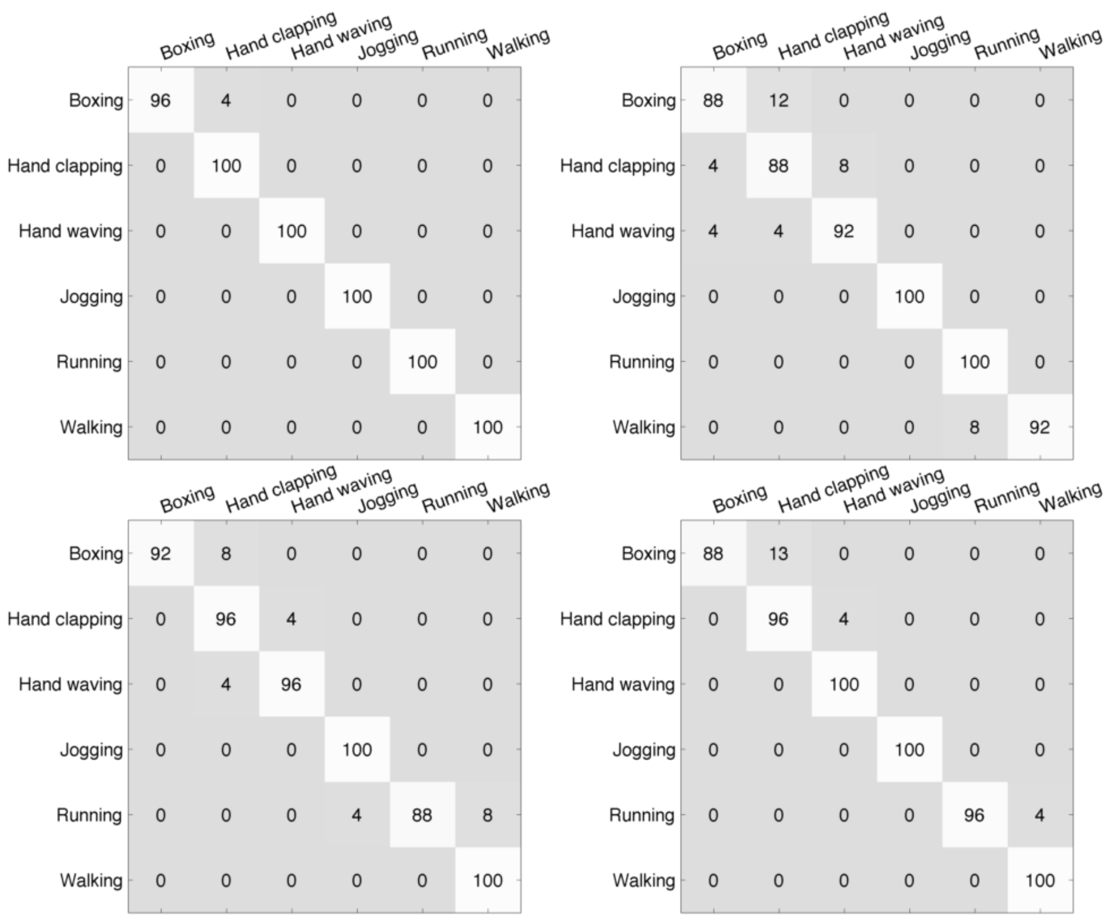
For labelling each video sequence, the proposed method takes about 0.027 seconds, while the Algorithm in Lui *et al.* (2010) and Algorithm in Lui and Beveridge (2011) take 0.297 and 3.669 seconds respectively.

	Scene 1	Scene 2	Scene 3	Scene 4
Algorithm in Lui *et al.* (2010)	98.67%	**98.00**%	95.87%	97.33%
Algorithm in Lui and Beveridge (2011)	98.67%	**98.00**%	**96.67**%	**98.67**%
Proposed	**99.33**%	93.33%	95.33%	96.67%

Tab. 4.2. Recognition results of KTH action data set.

4.5 Summary

In this chapter, we have investigated the low dimensional structure of the geometrical representation of the video sequences. This idea is motivated by the fact that the dimensionality of the Grassmannian manifold used in practice is very high, and the action data points tend to lie in a low dimensional space. The approach we proposed is based on the principal



	Boxing	Hand clapping	Hand waving	Jogging	Running	Walking
Boxing	96	4	0	0	0	0
Hand clapping	0	100	0	0	0	0
Hand waving	0	0	100	0	0	0
Jogging	0	0	0	100	0	0
Running	0	0	0	0	100	0
Walking	0	0	0	0	0	100

	Boxing	Hand clapping	Hand waving	Jogging	Running	Walking
Boxing	88	12	0	0	0	0
Hand clapping	4	88	8	0	0	0
Hand waving	4	4	92	0	0	0
Jogging	0	0	0	100	0	0
Running	0	0	0	0	100	0
Walking	0	0	0	0	8	92

	Boxing	Hand clapping	Hand waving	Jogging	Running	Walking
Boxing	92	8	0	0	0	0
Hand clapping	0	96	4	0	0	0
Hand waving	0	4	96	0	0	0
Jogging	0	0	0	100	0	0
Running	0	0	0	4	88	8
Walking	0	0	0	0	0	100

	Boxing	Hand clapping	Hand waving	Jogging	Running	Walking
Boxing	88	13	0	0	0	0
Hand clapping	0	96	4	0	0	0
Hand waving	0	0	100	0	0	0
Jogging	0	0	0	100	0	0
Running	0	0	0	0	96	4
Walking	0	0	0	0	0	100

Fig. 4.15. Confusion matrix of the proposed method on the KTH human action data set. The first row shows that results on scenarios 1 and scenarios 2 respectively, and the second row shows that results on scenarios 3 and scenarios 4 respectively.

geodesic analysis method. We apply this framework on the Grassmannian manifold, and the query video is classified based on the information obtained from *PGA*. We compare the proposed method with previous algorithms based on the same geometrical assumptions. The experimental results show that the proposed approach has comparable accuracy in action recognition as previous work and, in contrast to previous similar methods, has a linear computational complexity in the sense of the number of action classes regardless of the number of training data points.

Part III

Exploring geometric structures for hashing methods

Chapter 5

NOKMeans: Non-Orthogonal K-means Hashing

Note: Some portions of this chapter are taken from Fu *et al.* (2014).

Hashing based approaches for approximate nearest neighbour search problems have an intimate relationship with manifold learning since both aim to learn a set of data points in another space. In this chapter, we give a brief introduction to hashing methods. We introduce a new technique called *NOKMeans*, which is a generalisation of the quantisation error-based hashing methods such as *ITQ* (Gong and Lazebnik, 2011) and *OKMeans* (Norouzi and Fleet, 2013). Our main contribution is to increase the encoding capability of the hashing functions while, at the same time, inheriting the merits of the quantisation error based methods. We have conducted experiments on several large scale data sets, and the performance of *NOKMeans* shows that it is superior in most cases to other state of the art techniques.

5.1 Introduction

Finding nearest neighbour points is a computational task which arises frequently in computer vision, information retrieval and machine learning. Naive search is infeasible for massive real-word data sets since it takes linear time to retrieve the nearest point in the data set. For low dimensional data, the nearest neighbour search problem can be addressed by efficient and effective tree based approaches such as KD-tree (Bentley, 1975) and R-tree (Guttman, 1984). Due to the curse of dimensionality, when it comes to the high dimensional scenario, tree based approaches deteriorate rapidly and can even become worse than naive search (Weber *et al.*, 1998). Recent years have witnessed a surge of interest in methods based on hashing which can return approximate nearest neighbour (ANN) data points. Hashing techniques have been utilised in large scale medical image searching (Yu, Zhang, Liu, Zhong, and Metaxas, 2013), image matching for 3D reconstruction (Cheng, Leng, Wu, Cui, and Lu, 2014), collaborative filtering for Recommender Systems (Shrivastava and Li, 2014) and approximating the SVM kernel (Mu, Hua, Fan, and Chang, 2014).

Locality sensitive hashing (*LSH*) (Indyk and Motwani, 1998) introduced the exploration of hashing methods that encode the data into binary codes. In *LSH*, each binary code is obtained by a projection matrix which is randomly generated. The randomness of the projection matrix guarantees that the similarity between data points is inherited into the binary codes if sufficiently many bits are used to encode the data points. In recent investigations, *LSH* has been used for the nearest neighbour search problem with the L_p norm (Datar, Immorlica, Indyk, and Mirrokni, 2004), the learned Mahalanobis metric (Jain, Kulis, and Grauman, 2008), the Reproducing kernel Hilbert space (Kulis and Grauman, 2009) and search in the sense of maximum inner product (Shrivastava and Li, 2014). All of these algorithms belong to the category of *data independent hashing algorithms* whose main feature is that the binary codes are obtained from some random matrices. Although their performance is guaranteed, in order to find the nearest neighbour points effectively, a large number of bits are required which causes efficiency and memory problems in practical systems.

Another category of hashing algorithms attempt to encode data points by utilising the distribution information which is obtained from the training data set. Since the bits are used effectively, compact binary codes obtained from data dependent methods often return more satisfactory neighbour points. One of the seminal approaches in this vein is *Spectral Hashing* (Weiss, Torralba, and Fergus, 2008), which calculates binary codes by embedding the data points into Hamming space. The optimal embedding is determined by the following characteristics: neighbourhood relationships should be preserved, the code should be balanced (-1's and 1's should occur with roughly equal frequency), and bits should be pairwise independent. This hashing approach has been extensively developed in recent years (Wang, Kumar, and Chang, 2010b; Liu, Wang, Kumar, and Chang, 2011; Liu, Mu, Kumar, and Chang, 2014; Xu, Bu, Lin, Chen, He, and Cai, 2013). In some data sets, label information is available. This motivates hashing approaches that utilise label information of the data points. For example, if two data points have the same label, the similarity between them can be assigned as 1, and -1 otherwise. In Supervised Hashing (Liu, Wang, Ji, Jiang, and Chang, 2012), the objective function aims to learn binary vectors such that the $cosine$ similarity between them are the same as the similarities summarised from label information. In minimal loss hashing (Norouzi and Blei, 2011), a similar similarity matrix is used to construct a loss function, and the solution of the resulting optimisation problem is obtained by solving a structural SVM problem.

Another approach to the design of binary codes is to minimise the quantisation error between the binary codes and the original data points directly. Intuitively, reducing quantisation error means that less information is lost during the encoding. Since these algorithms use a binary encoding, the input space is divided into two pieces for each bit. Most hashing algorithms address this kind of partition problem by learning a set of hyperplanes which can be viewed either in the original space or in the Reproducing Kernel Hilbert space (RKHS). Each data point is encoded by its relative position to these hyperplanes, -1 for one side of the plane, 1 for the other. The left picture in Fig. 5.1 shows a 2D space partitioned by two hyperplanes. When the hyperplanes are mutually orthogonal, we can find a set of data points (indexing centres) such that the partition of the space (Voronoi diagram) according to this set coincides with the space partitioned by the hyperplanes. The hash code for the indexing centres can be predefined since these points are chosen from the vertices of a hyper-cube (Gong and Lazebnik, 2011) or hyper-cuboid (Norouzi and Fleet, 2013). The middle picture in Fig. 5.1 shows such a set of indexing centres (red points). Thus the encoding process for each data point can also be viewed as assigning to it the same binary code as its nearest indexing centre. In *ITQ* (Gong and Lazebnik, 2011), the data points are first projected onto the *PCA* directions, and then rotated in such a way that the total quantisation error of the training data points is minimised. This quantisation error based approach has been further extended to Kmeans Hashing (He, Wen, and Sun, 2013) and *OKMeans* (Norouzi and Fleet, 2013).

The focus of this chapter is to design a compact binary code based on minimising quantisation error. We propose a novel hashing algorithm: Non-Orthogonal K-means (*NOK-Means*). The essential idea of this algorithm is that increasing the freedom of the separating hyperplanes can lead to a better binary code in the sense of retrieval performance. We achieve this by relaxing the orthogonality constraints in Gong and Lazebnik (2011) and Norouzi and Fleet (2013) to a near orthogonal assumption. One problem introduced by this relaxation is that the explicit indexing centres cannot be found in the original feature space any more. This is because for any given indexing centres, the space will be divided into Voronoi cells if we index each data point by its nearest indexing centre. When the hyperplanes are not mutually orthogonal, it is impossible to find centre points such that the hyperplanes are exactly the separating boundaries for the Voronoi diagram. We address this problem by viewing the data points in a re-represented space where we can find specific

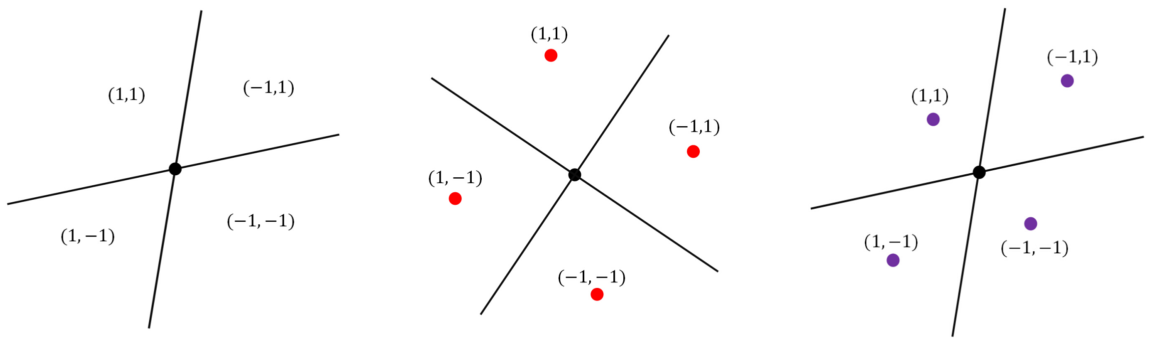


Fig. 5.1. Visualisation of different encoding approaches. The left figure shows two hyperplanes in general position. The space is divided into four pieces. The binary code of a data point in this space can be determined by its position relative to these hyperplanes. The middle figure shows that when the hyperplanes are mutually orthogonal, we can find a set of indexing centres (red points) whose binary codes have been predefined. The binary code of a point in this space can be determined by assigning it the same code as its nearest indexing centre. The right figure is a visualisation of our proposed method. The purple points are the implicit indexing centres which are viewed in the re-represented space.

indexing centres. After encoding each data point into its nearest indexing centre, the hyperplanes are exactly the separating boundary of the Voronoi diagram. In the right picture in Fig. 5.1, the purple points are viewed as a set of indexing points after re-representation.

5.2 Background

5.2.1 Notation

Suppose we want to build a binary code index for the data set $\{x_1, x_2, \cdots, x_N\}, x_i \in \mathbb{R}^D$. Denote $X \in \mathbb{R}^{D\times N}$ as the data matrix where each column is a data point. The binary code for this data set is denoted as $B \in \{-1,1\}^{d\times N}$ where each column corresponds to a d bit binary code in Hamming space. The Hamming matrix of equal size to X whose entries are ± 1 according to the sign of the corresponding entry of X is denoted $\mathrm{sign}(X)$. Assuming the data set is already centred and the hyperplanes pass through the origin, each hyperplane can be fixed by its normal direction. Therefore, the d hyperplanes, which are used to determine the binary code, can be specified by a hyperplane matrix (projection matrix) $A \in \mathbb{R}^{D\times d}$ where each column corresponds to one normal direction. Another view of the space which is partitioned by the hyperplanes is that the space is divided into regions (partitions) where the data points in the same region have the same binary code. We use **1** to represent an all ones column vector and I is the identity matrix. Denote k as the number of true nearest neighbour points for each query and K as the maximum number of data points retrieved in Hamming space.

5.2.2 Related work

When given a set of indexing centres, the encoding process shares some similarity with the K-means clustering algorithm. Each data point is encoded into a binary code according to its nearest indexing centre which behaves as a cluster centre in the K-means algorithm. *ITQ* (Gong and Lazebnik, 2011) aims to find an optimal binary code in the sense of minimal quantisation error. Specifically, the data points are preprocessed by centring and then projecting to a low dimensional space by *PCA*, and then solving an optimisation problem. The main idea of the optimisation problem is to find a rotation $R \in \mathbb{R}^{d \times d}$ in order to minimise the quantisation error between the rotated data points and their corresponding indexing centres. Suppose $V \in R^{d \times N}$ is the preprocessed data, and $B \in \{-1, 1\}^{d \times N}$ is the encoding binary matrix. The optimisation problem of *ITQ* is:

$$\begin{array}{rrcl} \min & J(R,B) & = & ||B - RV||_F^2 \\ s.t. & R'R & = & I \\ & B & \in & \{-1,1\}^{d \times N} \end{array} \tag{5.1}$$

If we combine the *PCA* projection matrix P and the final orthogonal matrix R together, we can see that *ITQ* has the following features:

- It aims to find a set of mutually orthogonal hyperplanes;
- Its optimisation model is to find minimum quantisation error in the subspace obtained from *PCA*;
- Its index centres are from the vertices of a rotated d-dimensional hyper-cube in the *PCA* subspace.

The left picture in Fig. 5.2 shows a visualisation of *ITQ*. Two hyperplanes are used to partition the data points. The red points are the corresponding indexing centres. The data points in the space can be encoded either by their position relative to the hyperplanes or by their nearest indexing centre.

OKMeans (Norouzi and Fleet, 2013) generalises *ITQ* by embedding the vertices of a d dimensional hyper-cube in $\mathbb{R}^D$, then the vertices are rotated by $R \in \mathbb{R}^{D \times d}$, scaled by $S \in \mathbb{R}^{d \times d}$ (S is a diagonal matrix) in the corresponding directions and translated by $\mu \in \mathbb{R}^D$.

Thus the indexing centres can be viewed as points chosen from the vertices of a rectangular hyper-cuboid. Finally, the optimisation objective is modelled as minimising the quantisation error which is formulated as:

$$\begin{aligned} \min \quad J(R, \mu, B, S) &= ||X - \mu \mathbf{1}' - RSB||_F^2 \\ s.t. \quad R'R &= I \\ B &\in \{-1, 1\}^{d \times N} \\ \mu &\in \mathbb{R}^D \\ S &\in \mathbb{R}^{d \times d}, \qquad S_{i,j} = 0 \text{ if } i \neq j \in \{1, 2, \cdots, d\} \end{aligned} \tag{5.2}$$

If we view $\mu \in \mathbb{R}^D$ as the 'origin' of the data points, the columns of the rotation matrix behave as the normal directions of the corresponding hyperplanes, the regions of the data points which have the same index are separated by these hyperplanes. A visualisation of *OKMeans* is shown in the right picture of Fig. 5.2. From the visualisation of both *ITQ* and *OKMeans*, we can see that both of the partitions from *ITQ* and *OKMeans* are divided by mutually orthogonal hyperplanes, and the indexing centres of *ITQ* and *OKMeans* are chosen from a unit hyper-cube and a rectangular hyper-cuboid respectively.

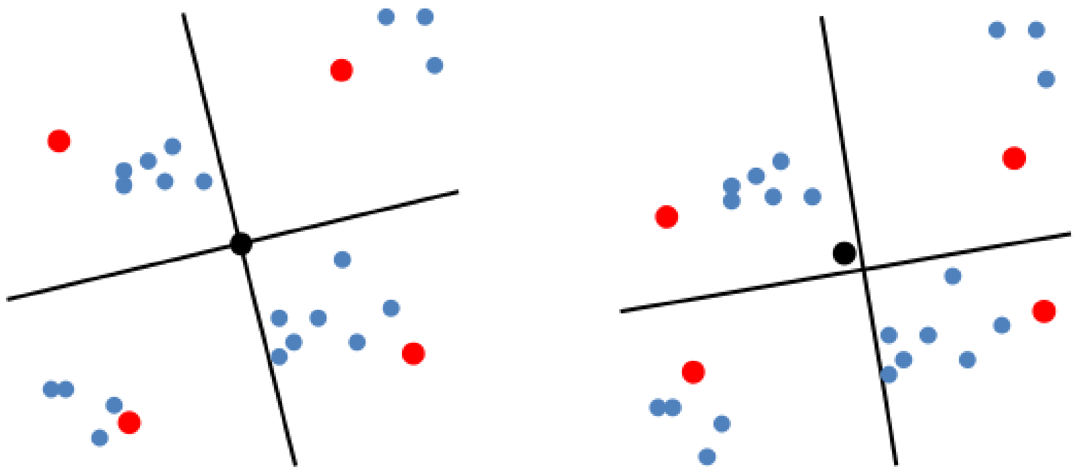


Fig. 5.2. Visualisation of data points encoded by *ITQ* and *OKMeans*. The black point is the centre of the data points. The red points are the indexing centres. The indexing centres are chosen from the vertices of a square (*ITQ*) and rectangle (*OKMeans*) respectively.

5.3 Algorithm

5.3.1 Motivation

As discussed above, the main idea of *ITQ* and *OKMeans* is to find a set of mutually orthogonal hyperplanes. The hyperplanes are used to partition the data points, thus we can index each data point as a point in $\{-1, 1\}^d$ according to its relative position to the hyperplanes. In this work, we investigate the situation when the orthogonality assumption used in *ITQ* and *OKMeans* is relaxed. Specifically, we adopt the 'near' mutually orthogonal property which is also used in Wang, Kumar, and Chang (2010a). One advantage of the relaxation is that it will increase the representation capability of the hyperplanes. The quality of the hash code is closely related to the position of the hyperplanes since the hyperplanes behave as the separating boundary of different index regions. Thus the flexibility of the position of the hyperplanes can lead to smaller overall quantisation error. Furthermore, as discussed in Wang *et al.* (2010a), the 'near' mutually orthogonal condition is favoured since the mutually orthogonal condition has some practical problems even though it is an approximation to the bit independent property. Therefore, to some degree, the near orthogonal constraint is a trade off between independence (Weiss *et al.*, 2008) and representation capability.

Besides, the assumption of a 'near' mutual orthogonal condition is consistent with the fact that the vectors in high dimensional space tend to 'near' mutual orthogonality with each other as the dimension increases. Consider two D dimensional random vectors $X = (x_1, x_2, \cdots, x_D)$ and $Y = (y_1, y_2, \cdots, y_D)$ where each coordinate is independently sampled from $\mathcal{N}(0, 1)$. The correlation between X and Y is defined as $\mathrm{Cor}(X, Y) = \frac{X'Y}{||X||||Y||}$. It is easy to verify that the expectation and variance of $\mathrm{Cor}(X, Y)$ is 0 and $\frac{1}{D}$ respectively. Thus, when D is large, random vectors tend to have the 'near' mutual orthogonal property which is one of the beneficial aspects of the curse of dimensionality. In Fig. 5.3, we have generated 64 random vectors in $\mathbb{R}^{128}$ and $\mathbb{R}^{960}$, and show their correlations. The left figure shows the correlation of the random vectors in $\mathbb{R}^{128}$. From this figure, we can see that most of the correlations are between -0.2 and 0.2. When considering the higher dimensional space, such as the correlation in $\mathbb{R}^{960}$ which is shown in the right picture, we noticed that the overall correlations tend to be much smaller.

When the separating hyperplanes are orthogonal to each other, we can find appropriate indexing centres in the original space. For example, if we view the red points in Fig. 5.2 as indexing centres, and then index each data point according to its nearest indexing centre, we will find that the region of different index partitions is exactly the same as the space partitioned by the hyperplanes. When the hyperplanes are not orthogonal, it is impossible to find indexing centres in the original space; when given a set of data points as indexing centres, the resulting segmented space will have a general Voronoi diagram structure and the hyperplanes will not coincide with the boundary diagram when the corresponding hyperplanes are not orthogonal. The left picture of Fig. 5.4 shows the resulting Voronoi cells for the indexing centres (purple points) in the original space. We can see that it is impossible to find two hyperplanes to separate these cells.

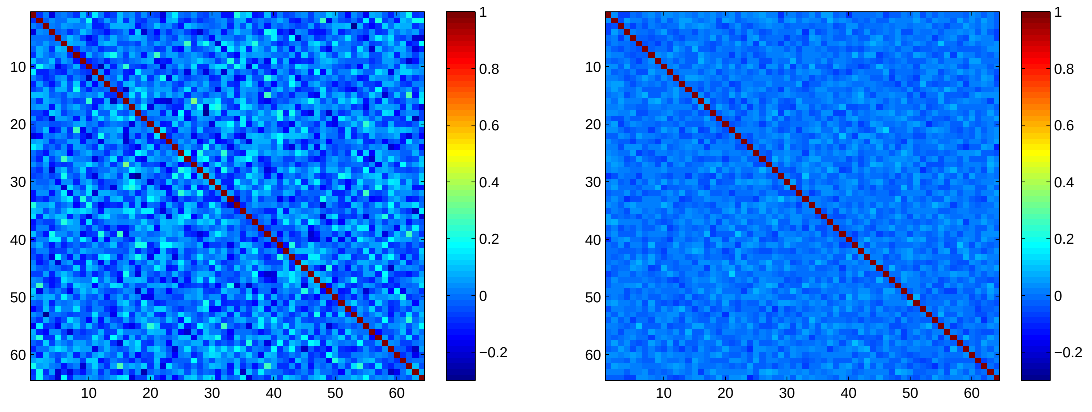


Fig. 5.3. Correlation of random vectors in $\mathbb{R}^{128}$ and $\mathbb{R}^{960}$ respectively. Each vector is obtained by sampling each coordinate from independent $\mathcal{N}(0,1)$ random variables, and then normalising each vector to have unit norm. From the picture, we can see that most of the correlations between the random vectors are small, and the overall correlations tend to be smaller when the dimension of the vectors increases.

In this work, we view the indexing centres in a transformed space used to re-represent the data points. In this way, if we index each data point by its nearest centre, the hyperplanes coincide with the boundaries of different index regions. The intuitive idea is as follows: suppose the hyperplanes are fixed, we can re-represent each data point by its relative distance to the hyperplanes. Fig. 5.5 shows one way to re-represent the data points. There are three hyperplanes in the original space. After re-representation, the data point is represented as a point in 3D space. In the new representation space, the vertices $\{-1,1\}^d$ can be viewed as the indexing centres. If we encode every data point as the binary code of its nearest indexing centre, we can see that the hyperplanes are exactly the boundary of the different indexing regions. An example of partitioning with non-orthogonal hyperplanes is shown in Fig. 5.4, the purple points can be viewed as the indexing centres in the new representation space. One advantage of this view is that the hyperplanes can be viewed

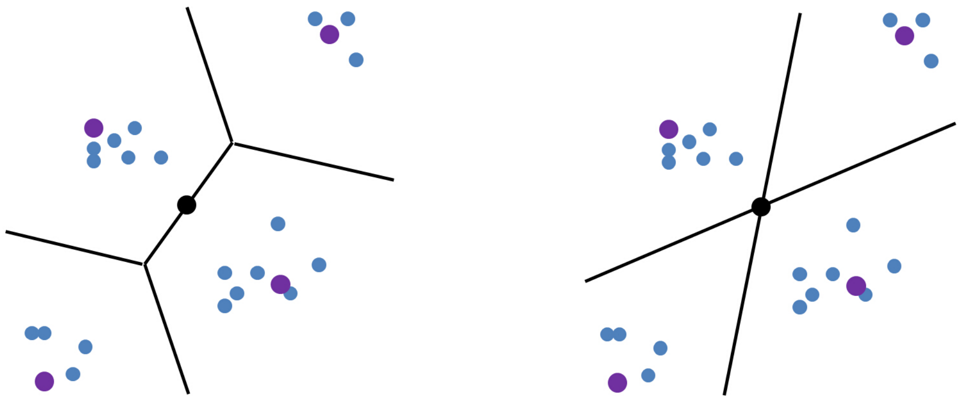


Fig. 5.4. Visualisation of the Voronoi diagrams under different views. The purple points in the left picture are viewed as the indexing centres in the original space. The resulting Voronoi cells can never be separated by two hyperplanes since the indexing centres are in general position rather than from the vertices of a square or rectangle. The right picture shows the resulting Voronoi cells when the purple points are viewed as the indexing centres in the re-represented space.

as the boundary of the different indexing regions. This re-representation trick allows us to formulate the problem by minimising the quantisation error between data points and indexing centres.

5.3.2 Formulation

When the orthogonal assumption is relaxed to near orthogonal, we propose the following optimisation model to learn the hyperplanes which can be used to index the data points:

$$\begin{array}{lrcl} \min & J(A,B) & = & \frac{1}{2N}||A'X - B||_F^2 + \frac{\lambda}{4}||A'A - I||_F^2 \\ s.t. & A & \in & \mathbb{R}^{D\times d} \\ & B & \in & \{-1,1\}^{d\times N} \end{array} \tag{5.3}$$

In above optimisation model, the columns of A behave as the normal directions of the hyperplanes, and $B \in \{-1,1\}^{d\times N}$ is the encoding matrix. When each column of A has a unit norm, the resulting re-represented data points are represented using the distance information of the data point to each hyperplane, i.e., each data point is represented by a vector where each element corresponds to the signed distance between the data point and one specific hyperplane. When the columns do not satisfy the unit norm property, the new representation can be viewed as a scaling effect on the data points represented

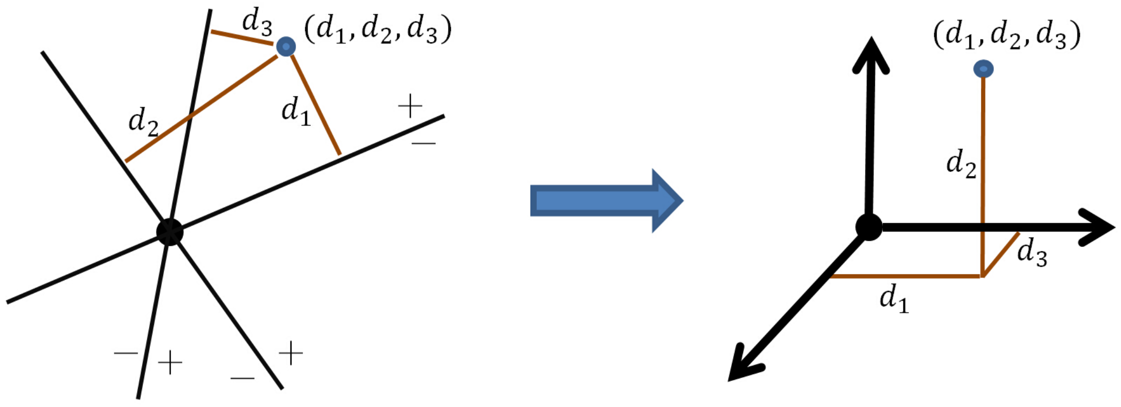


Fig. 5.5. Visualisation of re-representing the data point. The original two dimensional space is partitioned by three hyperplanes, and each data point is represented by its distance to the hyperplanes. We can see that each indexing region corresponds to one octant (there are 2^3 of them) in the right figure.

by distance information. The first part in the objective function is used to minimise the average quantisation error, the second part is used as a regulariser in order to maintain the near orthogonal property, λ is the regularisation parameter and the fraction is kept for convenience in further calculation.

For solving the above optimisation problem, we use alternating descent to find a locally optimal solution:

Fix A it is easy to see that $B = \text{sign}(A'X)$ is the optimal solution.

Fix B we use first order gradient descent to update the projection matrix A:

$$\frac{\partial J(A,B)}{\partial A} = \frac{1}{N} X(X'A - B) + \lambda(AA' - I)A \tag{5.4}$$

Therefore, A can be updated by $A - \gamma\frac{\partial J(A,B)}{\partial A}$. For the step length γ, we use a simple line search strategy, i.e. start with $\gamma = 1$, if the updated A does not improve the cost function, update the step length γ with $s\gamma$. The process is continued until the total cost is reduced.

In all of the following experiments, the initialisation of A is obtained by the *PCA* projection times a random rotation matrix which is a common approach used in hashing algorithms, and s is set to be 0.125. The backtracking line search process takes 10 to 20 iterations

Algorithm 6 Non-Orthogonal K-means

```
    Input: $N$ training data points $x_1, x_2, \cdots, x_N$, $I_{max}$ is the maximum iteration for the overall optimisation problem and $I_{step}$ is the maximum iteration for finding the step length.
    Output: The hyperplane matrix $A \in \mathbb{R}^{D\times d}$.
1: Center the training data points, denote the centred data points as $X \subset \mathbb{R}^{D\times N}$, and initialising the hyperplane matrix $A$.
2: for $i = 1$ to $I_{max}$ do
3:     Update $B$:
           $$B \leftarrow \text{sign}(XA).$$
4:     Calculate the gradient direction by Equation (5.4).
5:     Search the step length $\gamma$ by backtracking line search with at most $I_{step}$ iterations. If the maximum number of step length search iterations is reached, stop training stage.
6:     Update $A$ by:
           $$A \leftarrow A - \gamma \frac{\partial J(A, B)}{\partial A}.$$
7: end for
```

to find the appropriate step length. In order to make it feasible to run the algorithm in a reasonable amount of time so as to assess its output, we set the maximum iteration for line search step as $I_{step} = 50$. When the maximum number of iterations is reached, we stop the training stage since the projection is almost unchanged if we continue to update A by a very small step length. Finally, the maximum iteration I_{max} for the alternative descent method is set to 50 which is the same value used in *ITQ* and *OKMeans*.

5.3.3 Computational complexity

In each iteration, the time required to update B is $\mathcal{O}(NDd)$. The projection matrix A is updated in two steps. The first step is to calculate the gradient which takes $\mathcal{O}(ND^2 + NDd)$. Here $\mathcal{O}(ND^2)$ is used for calculating XX'. For establishing the step length, in each iteration, we have to check the appropriateness of the current γ, this involves calculating the objective function which takes $\mathcal{O}(NDd)$. Suppose the maximum iteration for the overall optimisation problem is I_{max} and the maximum iteration for finding the step length is I_{step}, thus the overall time required to update the hyperplane matrix A (Line 2 – Line 7) is $\mathcal{O}(I_{max}(NDd + ND^2 + NDd + I_{step}(NDd))) = \mathcal{O}(I_{max}ND^2 + I_{max}I_{step}NDd)$. The time required to calculate $X'X$, which can be precalculated, is $\mathcal{O}(ND^2)$. Therefore the overall computing time in training stage is $\mathcal{O}(ND^2 + I_{max}I_{step}NDd)$ which takes more computation than *ITQ* and *OKMeans* ($\mathcal{O}(I_{max}(Nd^2 + d^3)$ and ($\mathcal{O}(I_{max}(NDd + D^3))$ respectively).

During the training stage on our machine (implemented in MATLAB with single core), *NOKMeans* takes about 2.820 seconds per iteration when N, D and d are set to 10^5, 128 and 64 respectively, while for the same setting, *ITQ* and *OKMeans* take 0.259 and 0.419 respectively. This slowdown, by a factor of about 10, is typical for parameter values of N, D and d usually encountered in real problems. Since the line search step requires the most computation in the current implementation, the training stage can be sped up by choosing an appropriate initial step length γ and backtracking parameter s or a faster line search algorithm.

Finally, for encoding data and query points, it takes the same computational complexity as most hashing based algorithms. Each data point takes $\mathcal{O}(Dd)$ time to compute the binary code. For retrieving the K nearest neighbour points in Hamming distance, we use exhaustive search in all of our experiments since the distance calculation is efficient in Hamming space and it is easy to find the nearest neighbour points due to the distance property which only take values from $\{0, 1, 2, \cdots, d\}$. To speed up this process, one can build a hash table or use fast nearest neighbour searching designed for Hamming space (Norouzi, Punjani, and Fleet, 2012).

5.3.4 Discussion

The proposed method shares some similarity with *ITQ*. For *ITQ*, the final projection matrix is the *PCA* projection matrix P multiplied by the learned rotation matrix R, and, for the proposed algorithm, the projection matrix is learned during the optimisation process directly. When the parameter λ is infinite, the projection matrix A in *NOKMeans* shares the orthogonal property, i.e. $A'A = I$. Nevertheless there are still some differences between these two algorithms even as λ tends to infinity. For example, the quantisation error in *ITQ* is calculated in the *PCA* subspace, while the quantisation error in the proposed method is calculated in the space determined by A which is learned during the optimisation process.

Compared to *OKMeans*, the proposed algorithm also utilises a rectangular hyper-cuboid to some degree. When the parameter λ is positive, the learned projection matrix can be decomposed into $A = QS$ where each column in $Q \in \mathbb{R}^{D\times d}$ has a unit norm, and S is a diagonal matrix which has a scaling effect in each direction. Notice that the overall quantisation error $||AX - B||_F^2 = |S|^2||Q'X - S^{-1}B||_F^2$, thus our proposed method can be viewed as re-representing each data point by its distance information to each hyperplane, and then encoding the data point according to the indexing centres from the vertices of a hyper-cuboid. This hyper-cuboid is obtained by scaling the unit hyper-cube by S^{-1} in the corresponding directions. On the other hand, the S in the optimisation objective in *OKMeans* is viewed as an independent variable. One advantage of modelling the scale effect is that the resulting indexing centres will fit better to the data points in the sense of the quantisation error, but it also introduces a distortion problem. For example, the Hamming distance between neighbouring vertices is always 1, but their Euclidean distance is not fixed due to the scale effect of the hyper-cuboid in different directions. With the regularisation parameter λ in our method, the scale value will be constrained to around 1, and therefore the scale distortion is minimal.

5.4 Experiments

We evaluate the proposed hashing algorithm on four real data sets: SIFT1M (Jégou, Douze, and Schmid, 2011), SIFT10M, SIFT1B (Jégou, Tavenard, Douze, and Amsaleg, 2011), GIST1M (Jégou *et al.*, 2011), and compare the performance of related algorithms: *LSH* (Indyk and Motwani, 1998), *Spectral Hashing* (Weiss *et al.*, 2008), *ITQ* (Gong and Lazebnik, 2011), *OKMeans* (Norouzi and Fleet, 2013), on these data sets.

SIFT1M, SIFT1B, and GIST1M are three benchmark data sets which are used for testing the performance of different nearest neighbour searching algorithms. In SIFT1M and SIFT1B, each data point is a 128D SIFT feature (Lowe, 2004) extracted from Flickr images and INRIA Holidays images (Jégou, Douze, and Schmid, 2010). In GIST1M, each data point is a 960D GIST feature (Oliva and Torralba, 2001) which is extracted from the tiny image set (Torralba, Fergus, and Freeman, 2008), Holidays image set and Flickr1M (Jégou, Douze, and Schmid, 2008).

SIFT10M is our own data set. Each data point in this set is a SIFT feature which is extracted from Caltech-256 (Griffin, Holub, and Perona, 2007) by the open source VLFeat library (Vedaldi and Fulkerson, 2008). Caltech-256 is a benchmark image data set in computer vision that features a large number of classes (256) and high intra-class variations in each category. For each SIFT feature in the SIFT10M data set, we have the corresponding image patch which provides a kind of 'visualisation' of the corresponding SIFT feature and assists us to analyse the performance of different hashing methods. Fig. 5.6 shows some random collection of these patches. From the figure, we can see that the SIFT feature reflects the distribution information of the pixel value, such as the corner and boundary information, of the image in local region.

For SIFT10M, the base data points, training data points, and the query points are randomly chosen from all SIFT features extracted from the image set. The true nearest neighbour points are provided by exhaustive nearest neighbour search in 128D Euclidean space. For the other three benchmark data sets, we use the publicly available base points, training points, and query points, and true nearest neighbour information for the query points directly. Detailed information about these data sets including the number of training points, query points and base data points used in our experiments are summarised in Table 5.1.

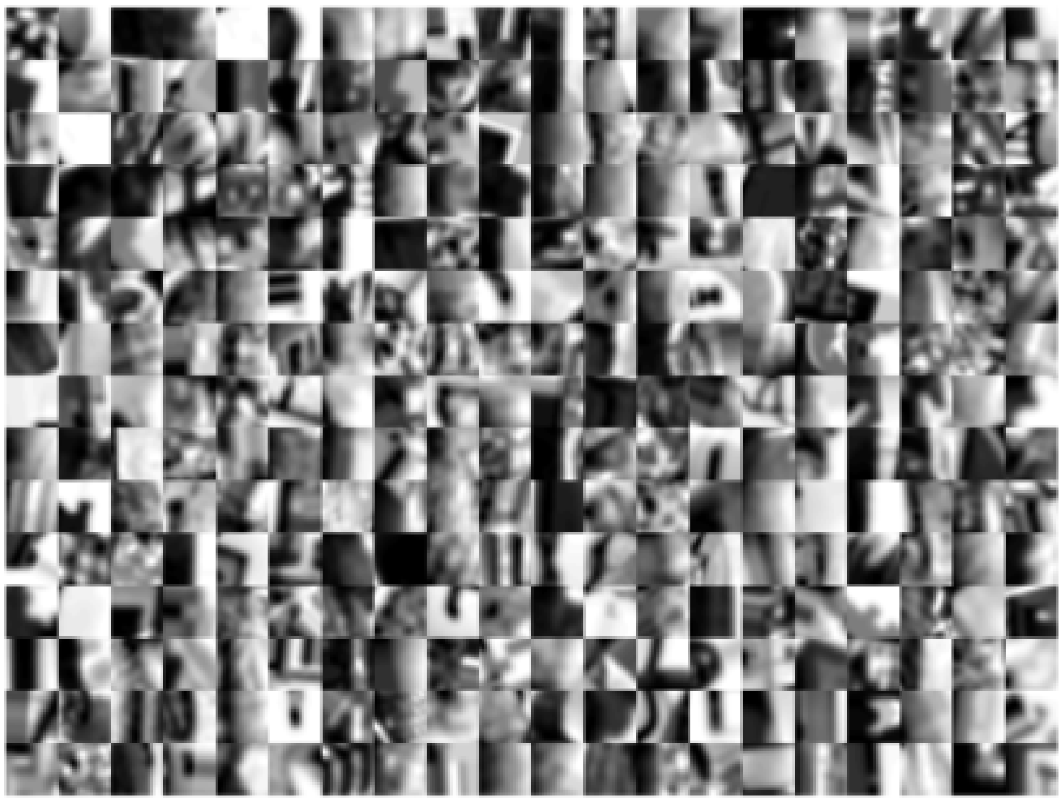


Fig. 5.6. Image patches associated with randomly chosen features from the SIFT10M data set. The SIFT features are extracted from Caltech-256 (Griffin *et al.*, 2007) by the open source VLFeat library (Vedaldi and Fulkerson, 2008). Each of the features is associated with a patch which is used for visualisation purposes. The key point detection method is often used to localise the interest regions, and then SIFT features are calculated. Due to the fact that boundaries or corners are important for representing images, from the figure, we can see that most of the patches correspond to this kind of information.

Data set	data type	dimension	base points	training points	query points
SIFT1M	SIFT feature	128	10^6	10^4	10^4
SIFT10M	SIFT feature	128	10^7	10^4	10^4
SIFT1B	SIFT feature	128	10^9	10^4	10^4
GIST1M	GIST feature	960	10^6	10^4	10^3

Tab. 5.1. Data sets which are used for evaluating different approximate nearest search algorithms. SIFT1M, SIFT1B and GIST1M are three benchmark data sets for testing the performance of hashing methods. SIFT10M is prepared for the purpose of visualisation as well as a medium size data set between SIFT1M and SIFT1B.

5.4.1 Performance measurements

For evaluating the performance of different hashing methods, we first investigate how the hashing methods are used for ANN problems. We take SIFT10M as an example since the corresponding patches provide the 'visualisation' of the retrieval results. Fig. 5.7 and Fig. 5.8 show two examples of finding the nearest neighbour data points for the query points respectively. The top left picture shows the patches of the query point and its true 99 nearest neighbour points among the $10,000,000$ data points. Here the patch of the query point is highlighted by a red border, and the patches of the true 99 nearest neighbour points are ordered by its next position of the same row or the next row if there is no space left in the same row. From top left picture, we can see that SIFT feature has a faithful representation of image patches since the corresponding patches of the neighbouring features are visually similar.

For each hashing method, we use 128 bits to encode each SIFT feature and retrieve $1,000$ nearest neighbour data points in sense of the Hamming distance. Here we should note that the $1,000$ data points might not be exactly the same as the true $1,000$ nearest neighbour data points in sense of the Euclidean distance and we only retrieve 0.01% of the base data set. Then we retrieve the 99 nearest neighbour data points among the $1,000$ selected points in sense of the Euclidean distance, and the 99 data points are placed accordingly in each subfigure. If the data point is among the true 99 nearest neighbour points of the query in sense of the Euclidean distance, we highlight it by a blue border, otherwise the patch is plotted without a border.

From Fig. 5.7 and Fig. 5.8, we can see that although the hashing methods return approximate nearest neighbour data points, i.e., the retrieved 99 points are not exactly the same as the true 99 nearest data points, the returned data points are very close to the query data points since their corresponding patches have similar visual appearance. Thus, this is the main reason why ANN is useful in practice. On the other hand, although the hashing methods return satisfactory results, we hope it can return the true nearest neighbour data points in sense of Euclidean distance as frequently as possible since this would guarantee the reliability of image retrieval or feature matching. Thus, the percentage of the true nearest neighbour data points retrieved is an important indicator for evaluating the performance of different hashing methods. From the number of the blue borders in Fig. 5.7 and Fig. 5.8, we can see that most of the true nearest neighbour data points are retrieved when only 0.01% of the base data set is considered. According to this indication, for these two query points, *NOKMeans* has the leading performance since it has the highest percentage for retrieving the true 99 nearest neighbour data points.

Based on the desirable features of an ANN method in practice as outlined above, we adopt recall as the main indicator for evaluating retrieval performance. For each query data point, we retrieve its nearest i data points in sense of the Hamming distance. Recall@i is the percentage of true nearest neighbour points in the retrieved data set, i.e.:

$$\text{Recall@}i = \frac{\#\text{retrieved true nearest neighbour points}}{\#\text{true nearest neighbour points}} \tag{5.5}$$

We have varied i from 1 to $K = 10,000$ to give a better overall picture of performance. Recall is an important indicator of retrieval performance and it also has internal relationships with other common measurements. For instance, higher recall often corresponds to higher precision. The main reason we use recall is that an ANN problem often involves a large scale data set, so K is large even if 1% of the base data set is retrieved, whereas k is much smaller. Thus, the precision does not fully reflect performance since it reduces to almost 0 when i increases.

In order to evaluate overall retrieval performance of the hashing algorithm, we use the m-Recall (mean recall) measure which averages Recall@i when the number of data points retrieved in Hamming space is up to K:

$$\text{m-Recall} = \frac{\sum_{i=1}^{K} \text{Recall@}i}{K} \tag{5.6}$$

here K is the maximum retrieved nearest points in Hamming space.

From the definition of m-Recall, we can see that it mainly addresses retrieval performance when part of the data set is retrieved. The average over i reflects a kind of global information similar to mean average precision (MAP) which is calculated by averaging the precision and is commonly used in information retrieval. Actually, the only difference of

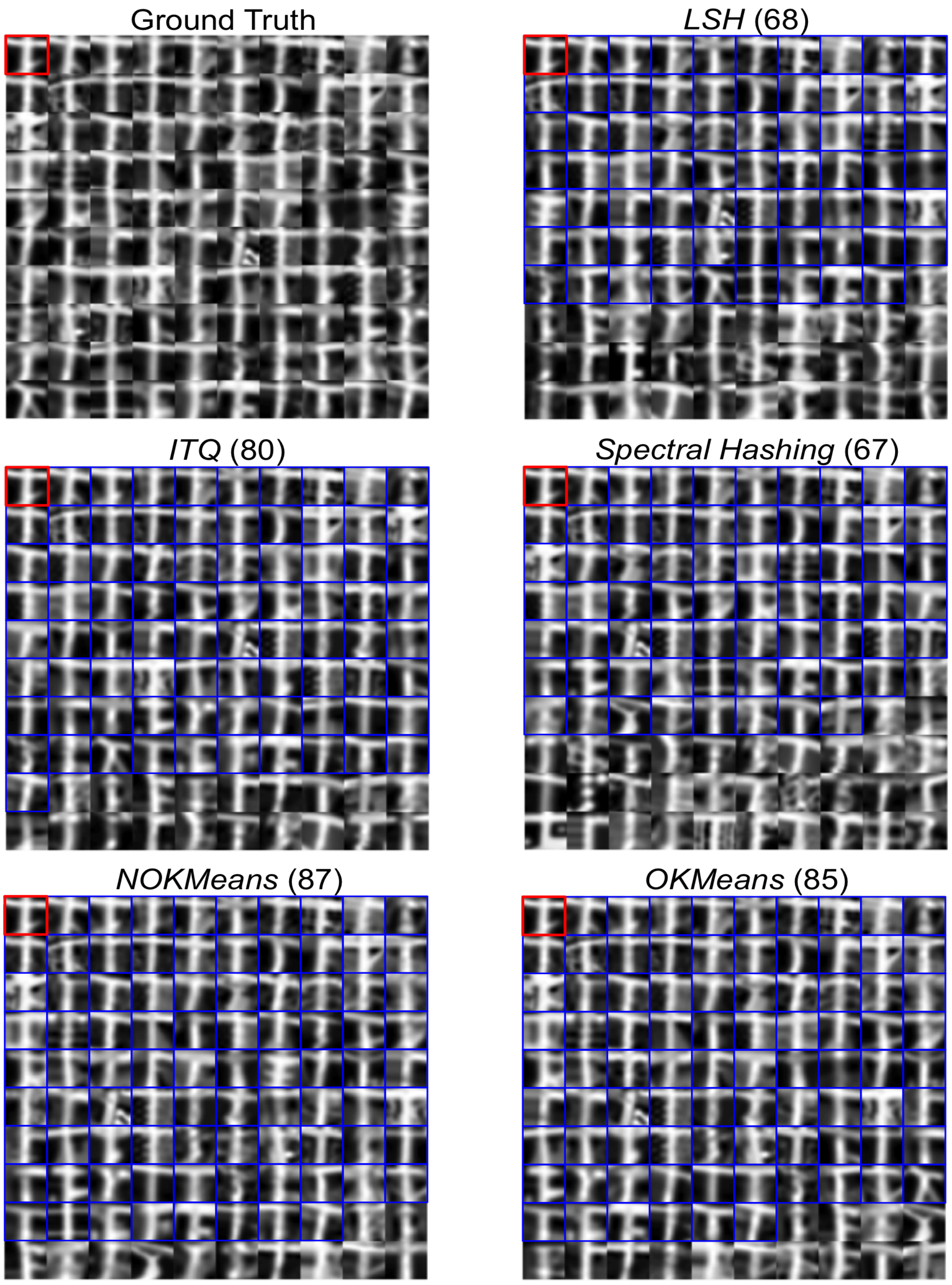


Fig. 5.7. Query results of the SIFT10M data set (1). The top left picture shows the patches of the query point and its true 99 nearest neighbour data points in sense of Euclidean distance. The other pictures show the retrieval results of different hashing methods. Each data point is encoded into 128D binary codes, 0.01% of base data is retrieved, and from among those the 99 nearest points in sense of the Euclidean distance are returned. The patches with blue borders are among the true 99 nearest neighbour data points, and the number of blue squares is shown in the brackets.

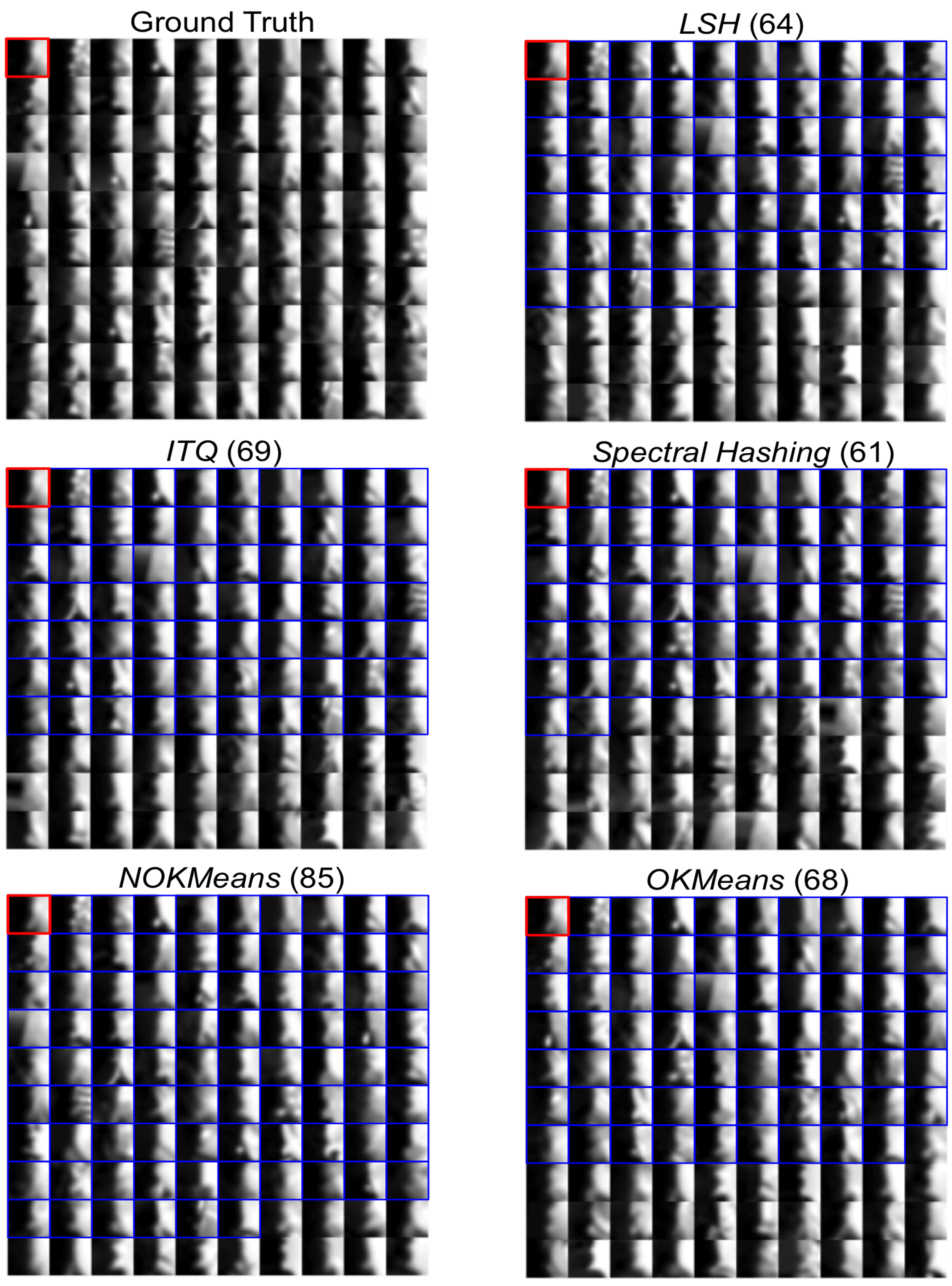


Fig. 5.8. Another Query results of the SIFT10M data set (2). The top left picture shows the patches of the query point and its true 99 nearest neighbour data points in sense of Euclidean distance. The other pictures show the retrieval results of different hashing methods. Each data point is encoded into 128D binary codes, 0.01% of base data is retrieved, and from among those the 99 nearest points in sense of the Euclidean distance are returned. The patches with blue borders are among the true 99 nearest neighbour data points, and the number of blue squares in shown in the brackets.

the definitions between recall and precision is the denominator, which is the number of true nearest data points for recall and the number of retrieved data points in precision. From the definitions, we can see that recall emphasises the percentage of true k nearest neighbour points is retrieved, while precision emphasises the percentage of true nearest neighbour points among the retrieved data points. As discussed previously, the k is much smaller than K for the ANN problem, the precision will be almost 0 due to the high value of K regardless of whether the true nearest neighbour points are retrieved or not. Besides, we can also view m-Recall as a weighted version of MAP if the weight w_i for Precision@i is defined as:

$$w_i = \begin{cases} \frac{1}{i}, & i = 1, 2, \cdots, K \\ 0, & i = K+1, K+2, \cdots, N \end{cases} \tag{5.7}$$

From the weights, we can see that m-Recall has a better reflection of the retrieval performance when a small part of the data set is retrieved. Thus, m-Recall is used here for evaluating the quality of the hashing bits.

5.4.2 Parameter selection

When using this model to learn the hyperplanes, we have to decide the value of the parameter λ. In order to test the sensitivity of the parameter λ in the optimisation, we choose λ from $\{10^1, 10^2, 10^3, 10^4, 10^5, 10^6, 10^7\}$. The m-Recall for different λ is shown in the right image in Fig. 5.9. From Fig. 5.9, we can see that, at beginning, the m-Recall increases as λ increases. After reaching the peak, the m-Recall dips moderately as λ keeps increasing. This motivates us to have a strategy to learn the index for different data sets. For each data set, λ is fixed by choosing from $\{10^1, 10^2, 10^3, 10^4, 10^5, 10^6, 10^7\}$ for one specific task, and choosing the parameter such that it reaches peak performance, then use this parameter for the whole data set. In the following experiments, the parameter is fixed as 10^4, 10^5, 10^6 and 10^5 for SIFT1M, SIFT10M, SIFT1B and GIST1M respectively. However it is worth noting that performance is similar for values of λ in the range 10^4–10^6 on all four data sets.

5.4.3 Results

We have evaluated *NOKMeans*, *ITQ*, *OKMeans*, *Spectral Hashing* and *LSH* on the four data sets. Fig. 5.10 shows the recall curves of different algorithms for searching the nearest neighbour point for each query. For each SIFT feature data set, we report the recall performance when the data points are encoded by $64, 96$ and 128 bits respectively. The experimental results enable us to view the performance of different algorithms from two dimen-

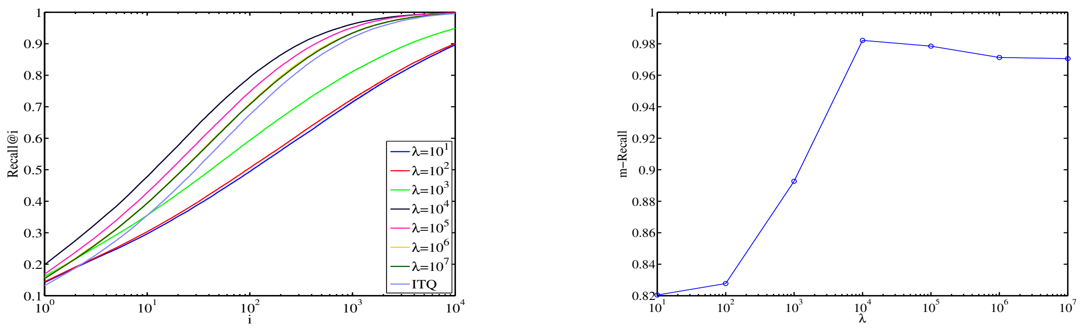


Fig. 5.9. Retrieval performance on the SIFT1M data set with different λ. Each data point is encoded into 128 bits, and the task is to retrieve the nearest neighbour point for each query. The performance of *ITQ* is also reported as a baseline, and its m-Recall is 0.9652.

sions, i.e. increase the number of bits to encode the SIFT feature and the scale of the data sets. From these two dimensions, we can see that, generally speaking, the more bits used to encode the feature points, the higher the recall. On the other hand, the nearest neighbour point searching problem becomes more difficult for bigger data sets. When each data point is indexed to a 64 bit binary code, the proposed algorithm has a comparable recall performance to the state of the art result (*OKMeans*), and has a much better recall performance than the remaining algorithms. When we use more bits to encode the data set, we find that the proposed algorithm has the highest performance among these algorithms.

The performance of the proposed algorithm coincides with our model assumption. This is because when few bits are used, the independence property, which leads to the mutually orthogonal condition, plays the main role when designing the hash code. When more bits are used to index the feature points, the mutual orthogonality condition leads to the loss of representation capability. Take the following toy example. Suppose the data points are distributed in a 2D subspace in 3 dimensional space, and we use three bits to encode the data points. If the three hyperplanes are mutually orthogonal, the data points are effectively encoded with two bits since the 2D subspace is partitioned into 4 different index regions. When the orthogonality condition is relaxed, the 2D space can be partitioned into 6 different indexing regions. So when the mutual orthogonality condition is relaxed, the separating capability of the hyperplanes will increase.

As discussed in Weiss, Fergus, and Torralba (2012) and He *et al.* (2013), retrieving different numbers of nearest neighbour points affects the performance of searching algorithms. Fig. 5.11 shows the recall curves of different algorithms for searching nearest neighbour

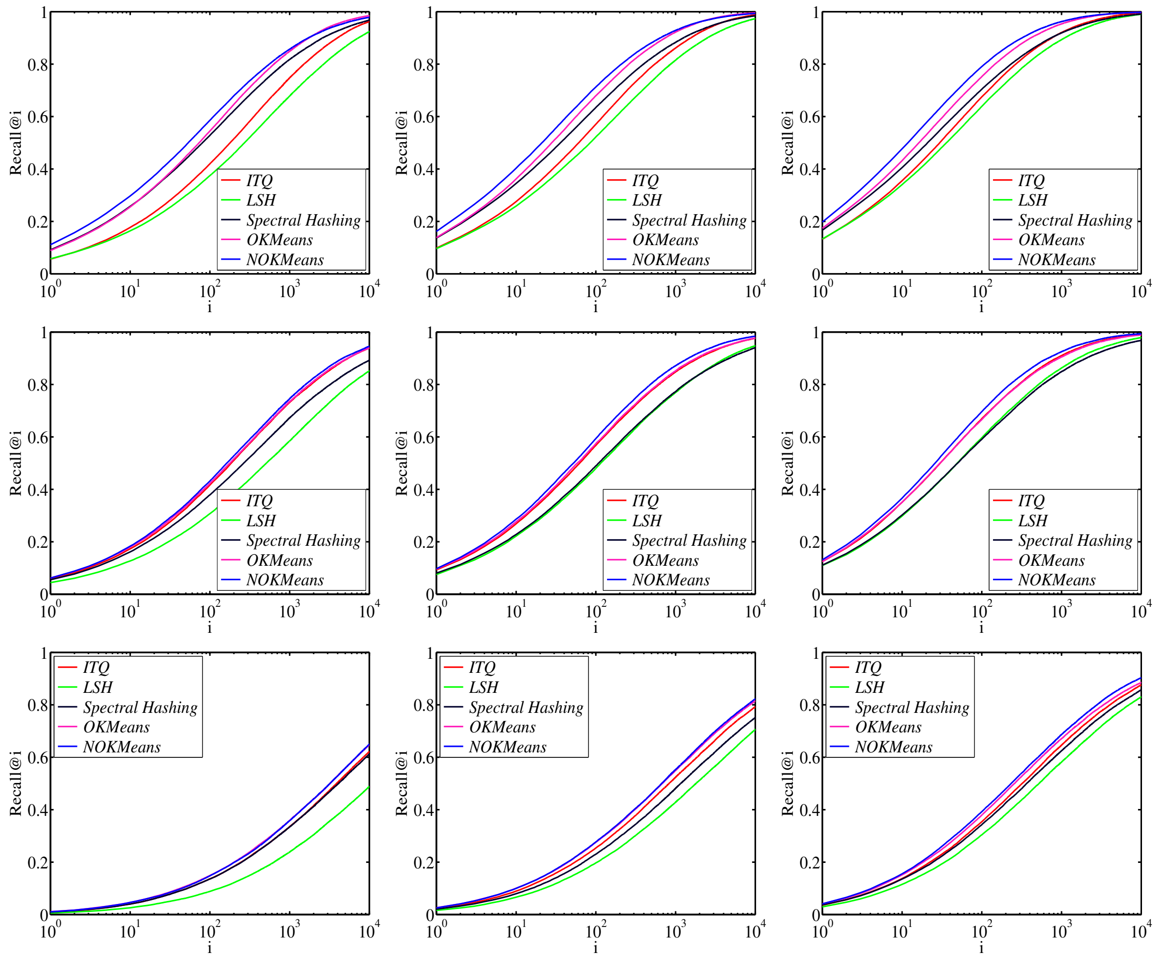


Fig. 5.10. Retrieval performance on the SIFT feature data sets. Different data sets are reported in different rows (SIFT1M, SIFT10M, SIFT1B respectively). For each SIFT feature data set, we show the Recall@i when each data point is encoded with $64, 96$ and 128 bits for columns $1, 2$ and 3 respectively.

points on SIFT1M. We report the recall performance when the data points are encoded by $64, 96$ and 128 bits according to the three columns respectively and k is set to be $1, 5, 10, 50$ and 100 according to the five rows respectively. As k increases, Recall@i decreases. Take $i = 1$ as an example, Recall@i can be either 0 or 1 when $k = 1$, whereas the Recall@i is at most 0.01 when $k = 100$. That is why Recall@i decreases in 5.11 for large k. Another reason for the decrease of the performance is that the i^{th} true nearest neighbour is more difficult to retrieve when i increases. Here the k true nearest neighbour points are ordered according to their distances to the query point. Thus, they are denoted as $1^{\text{st}}, 2^{\text{nd}}, \cdots, k^{\text{th}}$. Since the hyperplanes partition space into chunks, each data point belongs to one of the chunks. For the query data point, we can consider the chunk in which it is located. Now, it

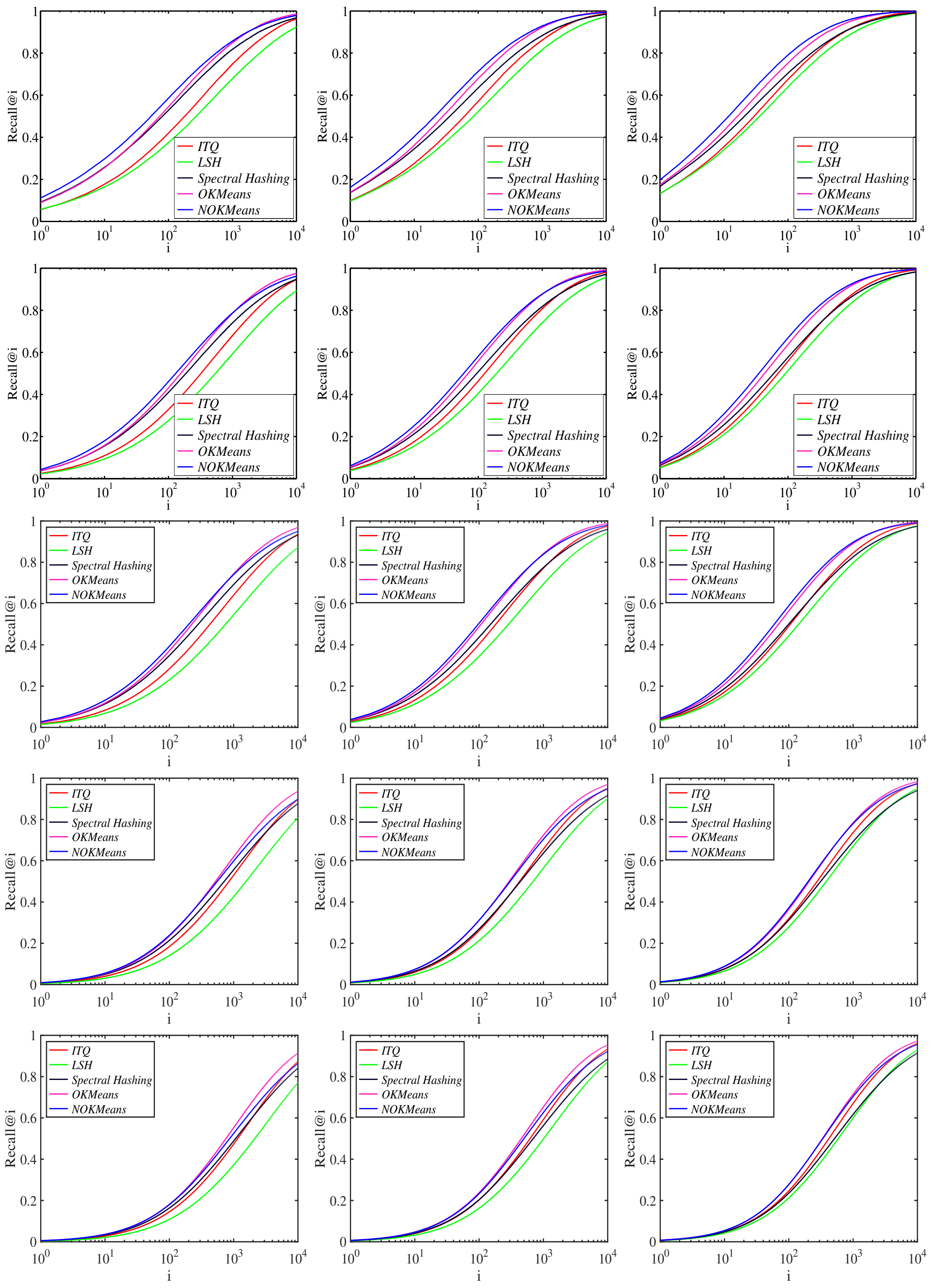


Fig. 5.11. Retrieval results on the SIFT1M data set for retrieving different k nearest neighbour points. Here, k ranges from $\{1, 5, 10, 50, 100\}$ for row $1, 2, 3, 4$ and 5 respectively, and each SIFT feature is encoded by 64, 96 and 128 bits for the columns $1, 2$ and 3 respectively.

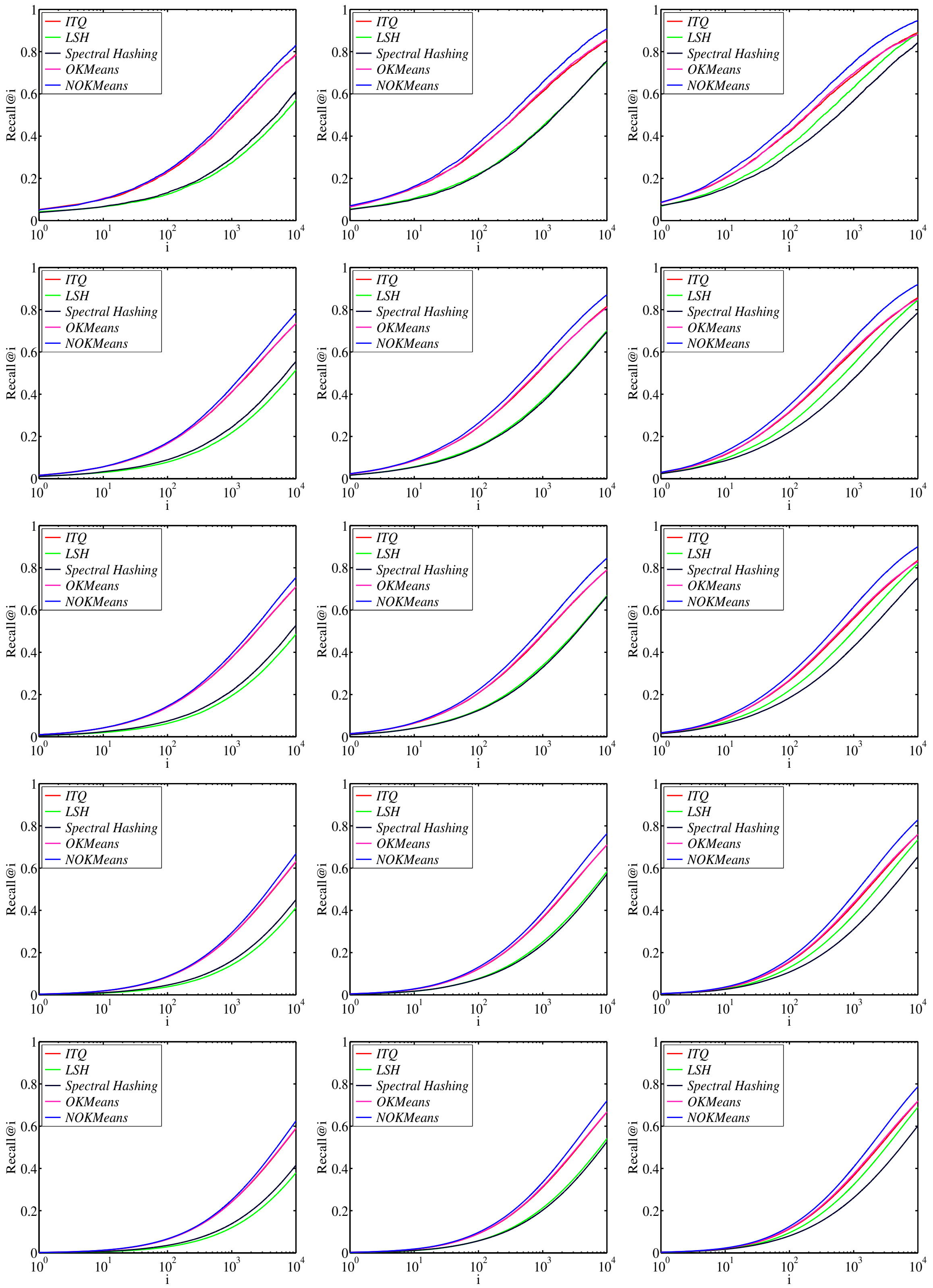


Fig. 5.12. Retrieval results on the GIST1M data set for retrieving different k nearest neighbour points. Here, k ranges from $\{1, 5, 10, 50, 100\}$ for row $1, 2, 3, 4$ and 5 respectively, and each SIFT feature is encoded by 64, 96 and 128 bits for the columns 1, 2 and 3 respectively.

is easy to prove that the i^{th} true nearest neighbour has smaller chance to be located in the same chunk when i increases.

From Fig. 5.11, we can see that *NOKMeans* has the leading performance when k is small. When k increases to 50 or 100, its performance degenerates as expected but it still has comparable performance to *OKMeans*. Besides, another phenomenon we can see from the figure is that *NOKMeans* has a good Recall@i when i is small such as up to $1,000$. These phenomena are partly due to the model we are using for learning the encoding functions. Since we increased the partitioning capability of the hyperplanes, they are more able to fit the data, and this is the main reason we can generalise the quantisation error based methods to have a better performance. But, at the same time, the fitness of the hyperplanes leads to degeneration more quickly when both k and K are large.

Fig. 5.12 shows the retrieval results on GIST1M data set, which has the highest number of dimensions among our test scenarios. We report the recall performance when the data points are encoded by 64, 128 and 256 bits according to the three columns respectively and k is set to be $1, 5, 10, 50$ and 100 according to the five rows respectively. Similar patterns as discussed in the SIFT1M data set are observed. From the figure, we can see that the performance gain of our proposed algorithm is larger when the number of encoding bits is increased and it has the leading performance across all of the different scenarios. Another phenomenon we can see from the figure is that, in order to retrieve more true nearest neighbour points, it is better to retrieve a relatively large number of points in the Hamming space. For example, if we retrieve $1,000$ nearest neighbour points in Hamming space, the recall of finding 100 nearest neighbour points is much lower than the recall of searching 1 nearest neighbour point, while for retrieving $10,000$ points in Hamming space, the recall gap between different nearest neighbour points is relatively small.

5.5 Summary

In this chapter, we have investigated a minimum quantisation error based hashing algorithm. Specifically, our focus is on the quantisation error of the re-represented data points. In this way, the Voronoi diagram in the original space is the same as the space which is separated by hyperplanes. Compared to the previous quantisation based algorithms, the hyperplanes learned in our algorithm are without the constraint that they are mutually orthogonal. We believe this relaxation leads to a better binary code index for large scale high dimensional data. We have tested the proposed algorithm on three benchmark data sets as well as a new SIFT data set. The experimental results show that our method performs better than current state of the art methods especially when it is used to encode high dimensional data points such as GIST1M.

Chapter 6

Auto-JacoBin: Auto-encoder Jacobian Binary Hashing

In the previous chapter, we discussed hashing-based approaches for approximate nearest neighbour search problems and presented our new method: *NOKMeans*, which is a generalisation of quantisation error based approaches. In this chapter, we investigate another way to keep geometric information about the data points when they are encoded into binary codes. Auto-encoder model is a two layer neural network which transforms the data set nonlinearly and keeps geometric information of the data set simultaneously. This motivates us to use an auto-encoder model to preserve the information during the binary encoding. At the same time, we find that the tangent spaces in local regions can be used to assist this purpose. Finally, we propose a new hashing method based on these observations: *Auto-JacoBin*.

6.1 Introduction

When high dimensional data points are indexed into compact binary codes, information loss is the main concern. Most previous work focuses on modelling the obtained binary codes optimally, i.e., the data points in the new space should have similar geometric information as in the original space. Although some algorithms including *ITQ* model the binary codes in the *PCA* subspace, these approaches still have the problem of poor geometry preservation due to the non-linear nature of binarisation.

In this work, motivated by the success that auto-encoder model has in preserving geometric information of the high dimensional data set (Van Der Maaten, Postma, and Van Den Herik, 2009; Hinton and Salakhutdinov, 2006), we are interested in investigating its ability to learn binary codes for the corresponding data points. Our optimisation model is constructed from both the auto-encoder model and the optimal binary code's perspectives.

For the auto-encoder model, we use a two layer network to keep geometric information consistent with the high dimensional data points. The main assumption of auto-encoder model is that the data points are sampled from some manifold. The task is to find a set of parameters for the neural network such that the data points from the manifold are reconstructible, i.e., the gap between the output data points and the input training data points is as small as possible. Points drawn from a domain around the manifold can be thought of as noisy data points. It is one of the features of the neural network model that it can process such points. Indeed, properly modelling the functions in this larger domain gives better feature learning capability for the model. Motivated by some recent efforts which aim to remove noise during the neural network forward propagation (Vincent, Larochelle, Lajoie, Bengio, and Manzagol, 2010; Rifai, Vincent, Muller, Glorot, and Bengio, 2011), we define the optimal noise removing function directly and investigate its first order approximation property which is used as a component in our later optimisation model. We believe that the high order information from the optimal noise removing function makes the learned auto-encoder model robust to noise, and enables the discovery of local relationships between the data points consistent with geometric information.

For the optimal binary codes' perspective, since our final interest is to obtain binary codes and the auto-encoder model does not provide these codes directly, the constraint, which minimises the gap between the learned features and the optimal ideal binary codes, is often used in designing hashing algorithms. We use an approximate 1-norm to model the gap minimising problem with the aim being to have a better distribution in Hamming space for the binary code.

Our main contributions are as follows:

- A novel "noise removing" function. We investigate a function which has a noise absorbing property, and find that its Jacobian matrix has an intimate relationship with the tangent space in the local region.

- A robust constraint that encourages the binary codes to have an optimal distribution in Hamming space.

- A novel hashing algorithm. The auto-encoder model is adapted for generating effective binary codes as well as preserving geometric relationships of the original real-world data sets.

6.2 Background

6.2.1 Notation

Suppose $X = [x_1, x_2, \cdots, x_N] \in \mathbb{R}^{D\times N}$ are the training data points, we denote the underlying distribution of the training data as the manifold $\mathcal{M}$, and, for each $x_i \in \mathcal{M}$, $T_i \in \mathbb{R}^{D\times d_{\mathcal{M}}}$ is a basis for the tangent space at x_i. We use the two layer auto-encoder model to ensure the hidden layer captures the geometric relationships in the training data set. Denote $W_1 \in \mathbb{R}^{d\times D}$ and $W_2 \in \mathbb{R}^{D\times d}$ as the weight matrices which connect the neural network from the input data points to hidden layer and the hidden layer to output layer, and the biases in the hidden and output layers are denoted as b_1 and b_2. The features from the hidden layer are denoted as $Y = [y_1, y_2, \cdots, y_N] \in \mathbb{R}^{d\times N}$ where $y_i = \tanh(W_1 x_i + b_1)$ and the features in the output layer are denoted as $Z = [z_1, z_2, \cdots, z_N] \in \mathbb{R}^{D\times N}$ where $z_i = \tanh(W_2 y_i + b_2)$. The main assumption of the auto-encoder model is that input and output should be as similar as possible. Thus the object of the auto-encoder model is to find a set of parameters for the neural network such that

$$\mathcal{J}_1(W_1, W_2, b_1, b_2) = \sum_{i=1}^{N} ||z_i - x_i||^2 \tag{6.1}$$

is minimised. The final binary code is obtained by $B = \text{sign}(W_1 X) \in \mathbb{R}^{d\times N}$.

6.2.2 Related work

The task of learning Hamming codes for a point cloud is a little different from the aim of auto-encoders used in other fields. This is because we not only hope to obtain binary codes, but also to use the optimal distribution in Hamming space: bit distributions that are equally likely and uncorrelated at each location (Weiss *et al.*, 2008). In this work, we introduce a constraint on the features in the hidden layer with the aim that the obtained binary codes have this optimal distribution. Due to the additional constraint, directly applying the auto-encoder model for the hashing problem does not provide us with many gains as measured by common retrieval indicators as shown in the experimental section. Thus, we have to explore an auto-encoder model which has the property that when the constraint is applied to the features in the hidden layer, the model still has the ability to transfer geometric information from the original data set to the features in hidden layer. This motivates us to seek a robust auto-encoder and adapt it for binary encoding.

For noise-free data, we expect that data points can be reconstructed exactly. Thus, it can be directly done by minimising $\mathcal{J}(W_1, W_2, b_1, b_2)$. For noisy data, it is not obvious that such a strategy is a good one. Considering a data point x out of the manifold $\mathcal{M}$ and z, the corresponding output. Training the auto-encoder model such that distance between x and z is minimised might deteriorate the final performance of auto-encoder model. In order to account for both the noise-free and noisy scenarios, we formulate the optimisation objective (Equation (6.1)) of auto-encoder model in a unified way. Denote f as a function which is defined one the manifold $\mathcal{M}$ and with some ϵ neighbourhood of it, and the restriction of f to $\mathcal{M}$ is the identity, i.e., $x = f(x)$ for $\forall x \in \mathcal{M}$. Thus the unified optimisation objective is to find a set of parameters for the neural network such that

$$\overline{\mathcal{J}}(W_1, W_2, b_1, b_2) = \sum_{i=1}^{N} ||z_i - f(x_i)||^2 \tag{6.2}$$

Note that for noise-free data, Equation (6.1) is equivalent to Equation (6.2), while for noisy data, the condition $f(x_i) = x_i$ might not be satisfied and we have to define f appropriately.

Recent efforts for dealing with noisy data are the denoising auto-encoders (*DAE*) (Vincent *et al.*, 2010) and contractive auto-encoders (*CAE*) (Rifai *et al.*, 2011). The main starting points of these algorithms are that the learned features in the hidden layer keep the important intrinsic structure of the original data set while unimportant information, such as the noise, is discarded as much as possible. In *DAE*, noise is manually added to the training data points; since we know the correspondence between the noisy and clean versions, a constraint is introduced to minimise the gap between these two versions according to some loss function. In *CAE*, the Jacobian norm of the function, which maps the input data points to the hidden layer, is minimised for a contractive effect. In the extreme case, *CAE* may contract all of the data points in the original space to a single point. This constraint is used to discard noise, but the distance between the input and output points is also considered. In the balance of these two constraints, data on the manifold will be unchanged and data outside the manifold will contract to the manifold.

Both *DAE* and *CAE* focus on training a model such that noisy data are projected onto the manifold and non-noisy data are left alone. When the data points have some noise, i.e., they are distributed around the manifold, output and the hidden layer should keep the important information and at the same time be robust to some degree of noise. This process is done implicitly with both approaches. *DAE* introduces artificial noise, and assumes that eliminating artificial noise will also help to eliminate real noise, i.e., the ideal noise data point is projected to the original data point through the forward propagation of the learned neural network. For *CAE*, the noise discarding process is done through balancing geometric consistency and the contractive property by minimising the norm of the Jacobian matrix. This motivates us to explore an explicit function which has the ideal noise absorbing property.

Fig. 6.1 displays one of the functions we are going to explore. Notice that each data point around the manifold is exactly projected to the nearest point in the manifold, while *DAE* and *CAE* do not have this kind of guarantee albeit they are designed to discarding the noise.

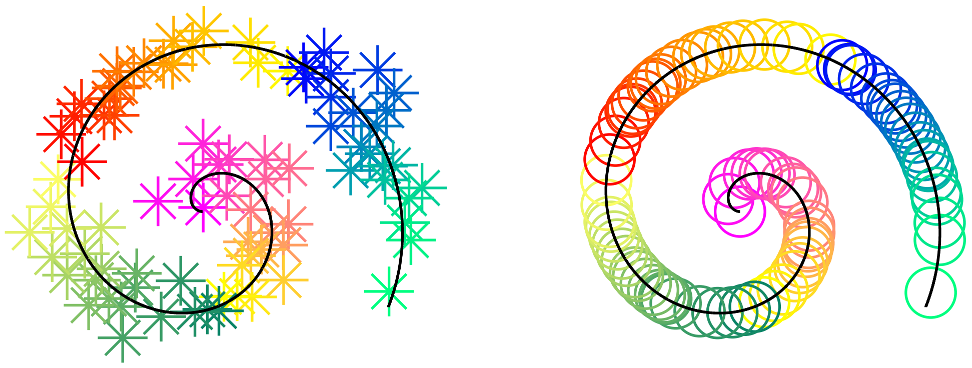


Fig. 6.1. The ideal function which projects data points near to the manifold onto their closest data points in the manifold. In the left figure, the coloured points, which are sampled around the black curved manifold, are the input data points for the ideal noise removing function. Here different data points are distinguished with different colours. The right figure shows the corresponding output data points of the function. Here each circle represents an output data point and corresponds to the star with the same colour in left figure.

6.3 Algorithm

6.3.1 Motivation

The binary codes we want to get have some intersections with the auto-encoder model. On one hand, the features of both approaches are from the hidden layer with the aim that features in the hidden layer have the same amount of information as in the original data set. On the other hand, previous auto-encoder models focus on obtaining ‘better’ features since their main purpose in learning these features is to train a classifier, and the auto-encoder model is viewed as a building block for a deep neural network. For hashing, we hope that the features in hidden layers are geometrically consistent with the original space, i.e., the relative order information between the data points is preserved, and the hidden features should be close to the binary codes under some metric.

Notice that both *CAE* and *DAE* try to project data points around the manifold to data points in the manifold. This motivates us to define a function f with this property directly, and the data points around the manifold can be viewed as noisy data points, i.e., it is generated by data points from the manifold plus some noise. Without any prior information we assume that each noisy data point is generated from the closest point on the manifold. Formally, we seek a function, f, such that

$$f(x) = \arg\min_{m \in \mathcal{M}} ||x - m||_F^2. \tag{6.3}$$

The object of training is to find a set of parameters for the network such that over all gap between the output points and the optimal recovered points is minimised.

The function f is only valid in some proper regions. This is because for data points outside the manifold, their closest point in the manifold might not be unique. In practice, we assume that the noise is constrained to a limited region around the manifold and therefore f is valid most of the time. Minimising the distance between x and z can be viewed as the 0^{th}-order approximation of the function f. More information about f can be captured by using higher order approximations and the Jacobian matrix of f is intimately related with the tangent space of $\mathcal{M}$:

Theorem 6.3.1. *Suppose $\mathcal{M}$ is a d dimensional compact smooth submanifold in $\mathbb{R}^D$, f is a function defined as $f(x) = \arg\min_{m \in \mathcal{M}} ||x - m||_F^2$, $\forall x \in \mathbb{R}^D$. For each m in $\mathcal{M}$, let $T_m \in \mathbb{R}^{D \times d}$ be the local normal basis of the tangent space to $\mathcal{M}$ at m. Then, the Jacobian matrix of f at m is $T_m T_m'$.*

Proof. To prove $J_m = T_m T_m'$, we have to show that

$$\lim_{t \to 0} \frac{||f(m+t) - f(m) - T_m T_m' t||}{||t||} = 0, \tag{6.4}$$

where $m \in \mathcal{M} \subset \mathbb{R}^D$ and $t \in \mathbb{R}^D$.

Since $m \in \mathcal{M}$, from the definition of f we can see that $f(m) = m$. Thus $f(m) + T_m T_m' t$ is the projection point of $m + t$ to the tangent space at m. Denote the positions of the points m, $m+t$, and $f(m+t)$ as O, A, and B respectively. Let C and D be the projections of A and B to the tangent space respectively, as shown in Fig. 6.2. Denote UV as the line segment between U and V, then $\overrightarrow{UV}$ and $||UV||$ are the corresponding vector and the length of the segment. Thus, Equation (6.4) is equivalently represented as:

$$\lim_{t \to 0} \frac{||CB||}{||OA||} = 0. \tag{6.5}$$

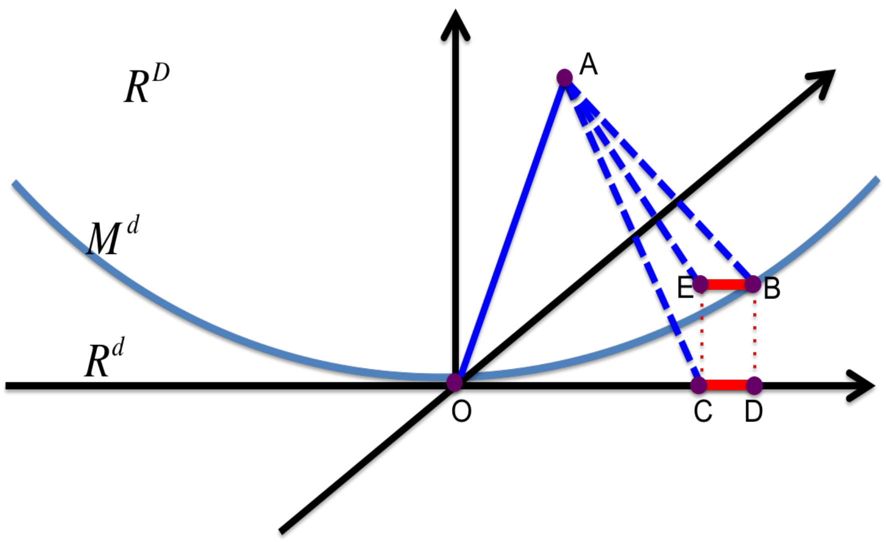


Fig. 6.2. Visualisation of the positions in Euclidean space. For better understanding, the blue curve can be imagined as a d dimensional manifold, and the x-axis is viewed as a tangent plane at position O.

In the following, we show that

$$\lim_{t \to 0} \frac{||CD||}{||OA||} = 0, \tag{6.6}$$

and

$$\lim_{t \to 0} \frac{||BD||}{||OA||} = 0. \tag{6.7}$$

Since CB is a side of the triangle CDB, we have $||CB|| \leq ||CD|| + ||DB||$. Thus, the results from Equation (6.6) and Equation (6.7) ensure that Equation (6.5) is true.

To prove Equation (6.6), we first locate the point E, such that $\overrightarrow{EB} = \overrightarrow{CD}$. Note that D is the projection point of B to the tangent space, and $\overrightarrow{CD}$ is a vector in the tangent space, thus we have $\overrightarrow{BD} \perp \overrightarrow{CD}$. It is easy to show that the quadrilateral EBDC is a rectangle. Thus, we can conclude that $\overrightarrow{EB} \perp \overrightarrow{EC}$. On the other hand, since C is the projection point of A in the tangent space, we have $\overrightarrow{AC} \perp \overrightarrow{CD}$. Therefore, we have $\overrightarrow{AC} \perp \overrightarrow{EB}$ since $\overrightarrow{EB} = \overrightarrow{CD}$. Considering the plane ACE, we have $\overrightarrow{AC} \perp \overrightarrow{EB}$ and $\overrightarrow{EC} \perp \overrightarrow{EB}$, therefore $\overrightarrow{EB} \perp$ the plane ACE and $\overrightarrow{EB} \perp \overrightarrow{AE}$.

Consider $\angle ABE$. B is the closest point on $\mathcal{M}$ to A, so

$$||OB|| \leq ||OA|| + ||AB|| \leq 2||OA||, \tag{6.8}$$

and $||OB|| \to 0$ when $t \to 0$. Therefore, the tangent space at B tends to the tangent space at O when $t \to 0$ since $\mathcal{M}$ is a smooth submanifold in R^D. From the definition of f, we can conclude that AB is perpendicular to T_B. Thus, $\angle ABE \to \frac{\pi}{2}$, when $t \to 0$. Since $\overrightarrow{EB} \perp \overrightarrow{AE}$, we have

$$||CD|| = ||EB|| = ||AB|| \cos \angle ABE \leq ||OA|| \cos \angle ABE. \tag{6.9}$$

The final inequality is true due to the definition of the function f. Thus we have

$$0 \leq \lim_{t \to 0} \frac{||CD||}{||OA||} \leq \lim_{t \to 0} \cos \angle ABE = 0, \tag{6.10}$$

and so Equation (6.6) is proved.

To prove Equation (6.7), we first show that

$$\lim_{t \to 0} \frac{||BD||}{||OD||} = 0 \tag{6.11}$$

We decompose the proof of (6.11) into two steps. The first step is to show it is true under special assumption when the direction of vector OD is fixed as t tends to 0. Considering the corresponding path (curve) of B, the tangent vector of this curve has the same direction as OD. Thus, we conclude that $\angle BOD \to 0$ when $t \to 0$. The second step is to show it is true for general case. Since $\mathcal{M}$ is smooth, the first order of the manifold structure is continuous. Thus we have $\angle BOD \to 0$ when $t \to 0$. Since $\frac{||BD||}{||OD||}$ is exactly the tangent of $\angle BOD$, we have the conclusion that $\lim_{t \to 0} \frac{||BD||}{||OD||} = 0$.

Since $\lim_{t \to 0} \frac{||CD||}{||OA||} = 0$ and $||OC|| \leq ||OA||$, $\forall \epsilon > 0$, $\exists \delta$, such that when $||t|| < \delta$, we have $0 \leq \frac{||OC||+||CD||}{||OA||} < 1 + \epsilon$. Therefore, when $||t|| < \delta$, we have

$$\begin{aligned} 0 \leq \lim_{t \to 0} \frac{||BD||}{||OA||} \leq & (1+\epsilon) \lim_{t \to 0} \frac{||BD||}{||OC|| + ||CD||} \\ \leq & (1+\epsilon) \lim_{t \to 0} \frac{||BD||}{||OD||} = 0, \end{aligned} \tag{6.12}$$

and so Equation (6.7) is proved. □

If we view the tangent space as a point in the Grassmannian manifold, Theorem 6.3.1 tells us that the Jacobian matrix at $m \in \mathcal{M}$ is exactly the point where the Grassmannian point T_m is embedded in $\mathbb{R}^{D \times D}$. Assuming the training data points are sampled from the manifold $\mathcal{M}$, for each point x_i, we estimate its tangent space T_i by the local *PCA* technique. So now, the manifold $\mathcal{M}$ is approximated by tangent patches at the training data points. On the other hand, when parameters of the three layer neural network are fixed, the output features can be viewed as a function of the input features. Thus, the Jacobian matrix of the function obtained from the neural network can be analytically expressed. Specifically, for the data point x_i, the Jacobian matrix J_i can be expressed as

$$J_i = W_1' \left(W_2' \odot \left(1 - z_2^2\right) \left(1 - z_3^2\right)' \right), \tag{6.13}$$

where $\odot$ is the point-wise product operator between two matrices and $(\cdot)^2$ is the element-wise square of the entries of a vector. Therefore, we propose to minimise the distance between J_i and $T_i T_i'$ in our optimisation model.

Since the features in the hidden layer range from -1 to 1, and binary codes are what we really need, we use the metric $||YY' - NI||_1$ to constrain the features in the hidden layer. The reason for this metric is that since the ideal elements in Y are either 1 or -1, the ideal diagonal position of YY' should be N exactly, and the non-diagonal position in YY' should be 0 due to the desirable property of uncorrelated binary codes. The 1-norm of $||\cdot||_1$ is calculated by summing the absolute value of the matrix's elements. Since the absolute value function is non-differentiable at 0, we approximate it by introducing a constant value $\epsilon = 0.0001$, i.e. $||a||_1 \approx \sqrt{a^2 + \epsilon}$. We denote the approximate 1-norm as $||\cdot||_1^\epsilon$.

Finally, the optimisation objective is

$$\begin{aligned} &\min C(W_1, W_2, b_1, b_2) \\ &= \sum_{i=1}^{N} (||x_i - z_i||_F^2 + ||J_i - T_i T_i'||_F^2) + \alpha ||YY' - NI||_1^\epsilon, \end{aligned} \tag{6.14}$$

where N is the number of training data points and α is a weight parameter balancing the optimal binary codes and the geometric relationships from the training data set.

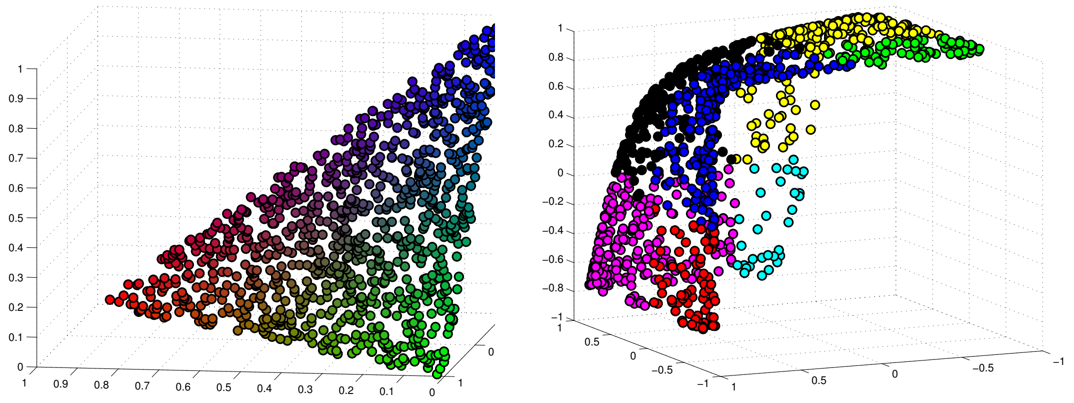


Fig. 6.3. Visualising of the warping. The left picture shows the training data set, which consists of $1,000$ data points sampled from $\{(x_1, x_2, x_3) \mid x_1 + x_2 + x_3 = 1 \text{ and } x_i > 0\}$. Each data point is distinguished by different colours. The right picture shows the hidden features. The colour of the data points are distinguished according to their positions in different quadrants. The neural network has the effect to warp the manifold of the data set into the surface of the cube in the 3D space.

Fig. 6.3 shows a toy example for the visualisation of the effects of the two optimisation components. The data points are randomly sampled from a plane $\{(x_1, x_2, x_3) \mid x_1 + x_2 + x_3 = 1 \text{ and } x_i > 0\}$. The left picture in Fig. 6.3 shows the training data points in the 3D Euclidean space. After optimisation, the hidden features are plotted in the right picture. Since we want to encode the data points into binary vectors, the data points are distinguished according their positions in different quadrants. The visualisation of the hidden features fully explains the motivation of the proposed optimisation model. Firstly, the robust auto-encoder is used to preserve the geometric information. For example, the transform from the training data points to the hidden features behaves like warping a piece of paper and the relative (distance) order information in local region is kept. Secondly, the constraint on the hidden features has the effect to disperse the hidden features to the vertices of the cube. Note that we only use three bits to encode the training data set, the number of different binary codes is at most 7. Thus, from the colour in the right picture, we can see that the constraint makes us use the binary codes fully. In all, the proposed optimisation model is capable of maintaining geometric information as well as learning optimal binary codes.

6.3.2 Optimisation

For optimising the objective function C, we propose to update the parameters by a mini-batch stochastic gradient descent method (Bengio, 2012). Since Equation (6.14) is non-convex, using mini-batch of training data achieves a better solution in general and this is the main reason we propose to use the stochastic gradient descent method for learning the parameters. The maximum number of epochs is set to be I_{max}. In each epoch, the training data is randomly divided into m mini-batches and the parameters are updated with each mini-batch of training data points respectively. For updating the parameters, we have to calculate the gradients: $\frac{\partial C}{\partial W_1}$, $\frac{\partial C}{\partial W_2}$, $\frac{\partial C}{\partial b_1}$ and $\frac{\partial C}{\partial b_2}$ respectively. The detailed gradient calculation is provided in appendix A. Denote θ_i as the vector of parameters W_1, W_2, b_1 and b_2 at the i^{th} iteration and G_i as the gradient of C at the i^{th} iteration. To find the step length λ_i, such that the Wolfe conditions are satisfied, we use the line search code developed by Mark Schmidt for Matlab [1].

Since $\tanh$ is used in the neural network, the data points have to be normalised properly. In our experiments, we scale the data points by the inverse of the maximum norm from the training data set, and then scale the data points by 0.8 again as the training data points might not fully cover the distribution of the base and query data sets. For initialising the

[1]`http://www.di.ens.fr/~mschmidt/Software/minFunc.html`

parameters, suppose μ is the mean of the training data points, W_1 is initialised by the *PCA* projection times a random rotation matrix, and W_2, b_1, and b_2 are initialised as W_1', $-W_1\mu$, and μ respectively. The reason that we use this set of initial parameters is that $\tanh$ can be approximated by a linear function around the origin, and this set of parameters is optimal for reconstruction error when a linear function is used in the neural network. For estimating the tangent planes, we retrieve the $(D + d)$ nearest data points for each x_i, and take the *PCA* projection which preserves 98% of the energy or the most energetic d dimensions, whichever is smaller. Algorithm 7 is a summary of the proposed training process.

Algorithm 7 Auto-JacoBin

```
    Input: Training data points $x_1, x_2, \cdots, x_N$, the maximum iteration $I_{max}$, and the number of batches $m$.
    Output: The parameters $W_1, W_2, b_1$ and $b_2$.
 1: Initialise $W_1, W_2, b_1, b_2$, and then vectorise them into a vector $\theta$.
 2: for $i = 1$ to $N$ do
 3:     Estimate the tangent space $T_{x_i}(\mathcal{M})$.
 4: end for
 5: for $i = 1$ to $I_{max}$ do
 6:     Randomly permute the order of the training data set, and divide the training data into $m$ subsets.
 7:     for $j = 1$ to $m$ do
 8:         Calculate the gradient $G_{ij}$ of the current $j^{\text{th}}$ subset.
 9:         Find the optimal step length $\lambda_{ij}$ such that the Wolfe conditions are satisfied by line-search.
10:         Update the parameter $\theta$:
                $\theta \leftarrow \theta - \lambda_{ij} G_{ij}$
11:     end for
12: end for
13: Reshape the vector $\theta$ into parameters $W_1, W_2, b_1$ and $b_2$.
```

6.3.3 Computational complexity

For the tangent plane estimation stage, the K nearest neighbours are located by brute-force which takes $\mathcal{O}(N^2K)$ computation time and the spectral decomposition takes $\mathcal{O}(ND^3)$.

This computation could be reduced by exploring the low dimensional structure such as the Nyström method (Drineas and Mahoney, 2005) but we have not done so yet.

In each iteration, calculating Y and Z takes $\mathcal{O}(NDd)$ time, and both the Jacobian matrices and the gradient calculation take $\mathcal{O}(ND^2d)$. Since we use a line search method to find the step length, it might take multiple evaluations to satisfy the Wolfe conditions. It is too computationally and memory expensive to use all training data points at each optimisation step, and therefore we approximate by using a batch size of $1,000$ points.

6.4 Experiments

To evaluate the performance of *Auto-JacoBin*, we conducted experiments on two benchmark image data sets and one local feature data set. Two data sets use global image features: 960D GIST features in the GIST1M (Jégou *et al.*, 2011) set; and 128D wavelet texture feature (Manjunath and Ma, 1996) for the NUS-WIDE (Chua, Tang, Hong, Li, Luo, and Zheng, 2009) set. We also use the local feature data set SIFT1M (Jégou *et al.*, 2011). For GIST1M and SIFT1M, the training set, query set and base set have the same setting as in the last chapter. For NUS-WIDE, we randomly choose $10,000$ points as the query points, $10,000$ points as training points, and the remaining $249,648$ data points as the base points.

The performance of *Auto-JacoBin* is compared with several state of the art hashing algorithms including *LSH* (Indyk and Motwani, 1998), *Spectral Hashing* (Weiss *et al.*, 2008), *ITQ* (Gong and Lazebnik, 2011), *OKMeans* (Norouzi and Fleet, 2013), and *NOKMeans* (Fu *et al.*, 2014). The performance of different hashing methods for real large scale high dimensional data sets is compared with Recall@i (Equation (5.5)) and m-Recall (Equation (5.6)) indicators.

6.4.1 Parameter selection

There are essentially two parameters that need to be chosen for our solution: the number of iterations of the line search optimisation process and the weight parameter α. We have analysed the costs during optimisation and notice that it converges quickly. As a case in point, Fig. 6.4 shows a typical cost changes during the optimisation stage. The experiment is conducted on the NUS-WIDE data set. The task is to learn 64 bits for encoding the base data set, and the weight parameter is set to be 0.1. From the figure, we can see that the objective function converges quickly and it makes tiny changes after the first 20 iterations. Another observation from the cost-iteration plot is that the cost decreases monotonically despite

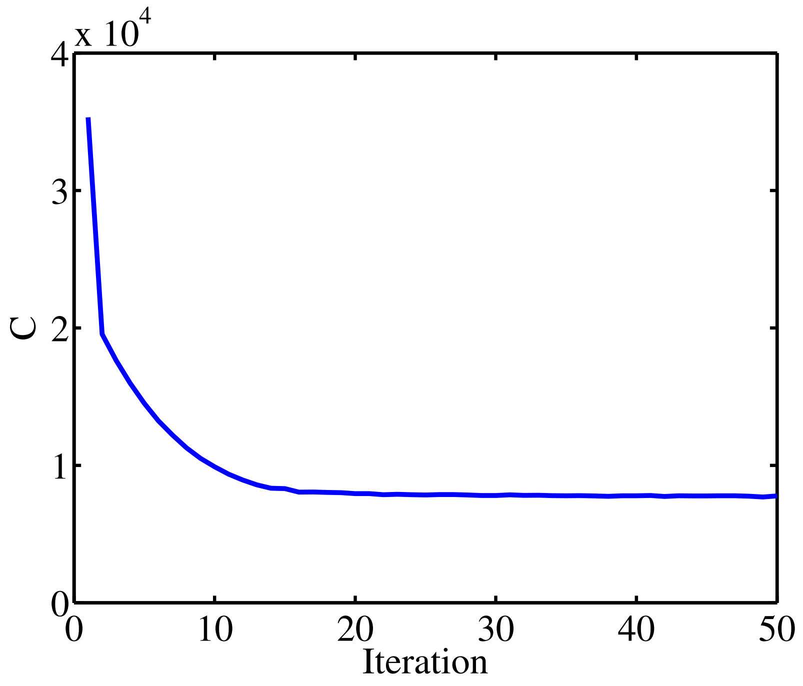


Fig. 6.4. Cost changes during the optimisation process. The experiment is conducted on NUS-WIDE data set with $d = 64$ and $\alpha = 0.1$. From the figure we can see that the optimisation converges quickly and there are almost no changes after the first 20 iterations.

lack of guarantees that this would occur since the input data points keep changing in each iteration. Thus, we can either set it to be a fixed value as it is done in our experiments or terminate when the cost is below a threshold. In our experiments, the number of training data points is $10,000$, the batch size is $1,000$, and the training data points are randomly shuffled in each epoch. Thus, each pass of the training data set takes $m = 10$ iterations. We set the total number of iterations to be 50 in accordance with other hashing methods. Therefore, the optimisation cycles through the training data set $I_{max} = 5$ times in total.

For the weight parameter α, we empirically test its value at $0.01, 0.1, 1, 10$. Fig. 6.5 shows the m-Recall when the data points are encoded with $64, 96$ and 128 bits respectively for the nearest point search for each query in the NUS-WIDE data set. For visual comparison, we have included the corresponding m-Recall performance of *NOKMeans* which is a competing hashing method as shown in the experiments later. The experiment is repeated 5 times. Since the m-Recall in each experiment is average over 10,000 query tests, the deviations of the m-Recall are relatively small for all of the tests. From the chart in Fig. 6.5, we can see that the proposed approach has stable performance when the weight parameter α ranges from 0.01 to 10, and its overall performance is always better than *NOKMeans* on the NUS-WIDE data set. In the following experiments, we fix the weight parameter α to be

0.1.

In order to show that the parameter $\alpha = 0.1$ is statistical significant, we adopt the two-sample t-tests between *Auto-JacoBin* with parameter $\alpha = 0.1$ and other methods in Fig. 6.5. The corresponding p-values are summarised in Table 6.1. According to the significance level $\alpha_t = 0.05$, *Auto-JacoBin* with parameter $\alpha = 0.1$ is statistical significant better than *Auto-JacoBin* with parameter $\alpha = 0.01, 10$ and *NOKMeans*. When the data points are encoded with 128 bits, *Auto-JacoBin* with parameter $\alpha = 0.1$ is statistical significant better than *Auto-JacoBin* with parameter $\alpha = 1$.

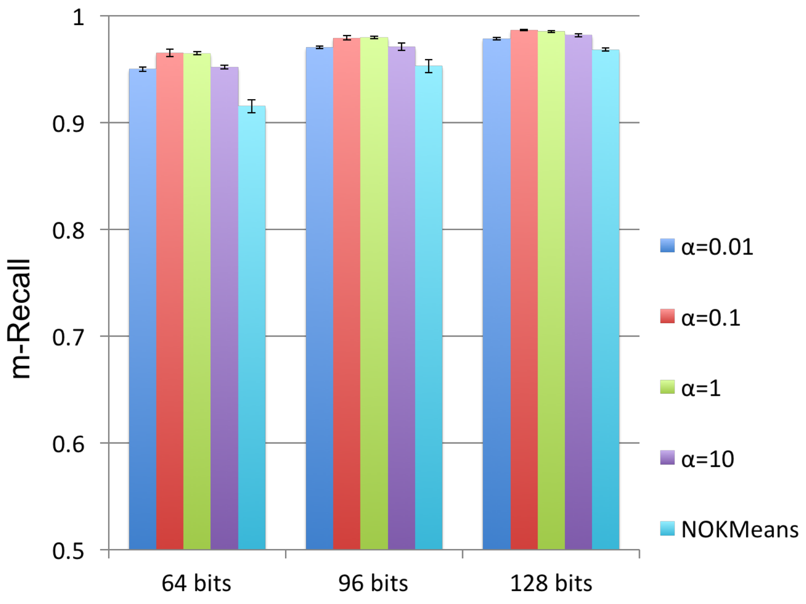


Fig. 6.5. The m-Recall performance when the weight parameter ranges from 0.1 to 10. The computational task is to find the nearest neighbour point for each query in the base data set of NUS-WIDE. The performance is compared when each data point is encoded into $64, 96$ and 128 bits respectively. The standard deviations of the m-Recall are shown in the corresponding bars.

6.4.2 Performance with different auto-encoder models

In order to see the impact of the first order constraint for *Auto-JacoBin*, we report the comparative results for AutoBin – an optimisation without the Jacobian component. The cost function of AutoBin is optimised with two different methods. One, referred to as AutoBin1, uses the same optimisation method as in *Auto-JacoBin*. The other, referred to as AutoBin2,

	$\alpha = 0.01$	$\alpha = 1$	$\alpha = 10$	*NOKMeans*
$\alpha = 0.1$ (64 bits)	3.22×10^{-5}	4.25×10^{-1}	9.78×10^{-5}	1.62×10^{-6}
$\alpha = 0.1$ (96 bits)	3.49×10^{-5}	5.81×10^{-1}	1.50×10^{-3}	1.40×10^{-4}
$\alpha = 0.1$ (128 bits)	1.94×10^{-5}	2.19×10^{-2}	6.01×10^{-4}	7.36×10^{-7}

Tab. 6.1. The p-values for two-sample t-tests between different methods. The null hypothesis is that the corresponding two methods have the same means of m-Recall but their variances might be different. The alternative hypothesis is that *Auto-JacoBin* with parameter $\alpha = 0.1$ has a larger mean of m-Recall.

is using the whole training data set in each iteration using LBFGS (Liu and Nocedal, 1989), and it only needs cost and gradient information for Mark Schmidt's Matlab toolbox.

The proposed auto-encoder is motivated from *DAE* and *CAE* which are designed to have a noise removing effect. It is also interesting to see that the performance of the corresponding hashing methods when the auto-encoder component in the proposed optimisation object is replaced with *DAE* and *CAE* respectively. We refer to these two new hashing methods as denoising auto-encoder binary hashing (DAutoBin) and contractive auto-encoder binary hashing (CAutoBin). For DAutoBin, the noise is injected by randomly setting a fixed percentage of elements as 0. Take the 960D GIST feature as an example, suppose we fix the percentage by threshold t, we generate a random number r_i ($i = 1, 2, \cdots, 960$) between 0 and 1 for each entry of $x = (x_1, x_2, \cdots, x_{960})' \in \mathbb{R}^{960}$, and set

$$x_i = \begin{cases} x_i, & r_i > t \\ 0, & r_i \le t \end{cases} \tag{6.15}$$

We empirically test the parameter t from a set $\{0.01, 0.05, 0.1, 0.2\}$ and the weight parameter α from a set $\{0.01, 0.1, 1, 10\}$. For CAutoBin, both the parameter λ for the Jacobian norm and the weight parameter α are tuned from the set $\{0.01, 0.1, 1, 10\}$. Finally, the optimisation for DAutoBin and CAutoBin is similar as Algorithm 7. The main difference is the gradient calculations which can be derived similarly as for *Auto-JacoBin*.

Fig. 6.6 shows the performance of the hashing methods based on different auto-encoder models in NUS-WIDE. Each data point is encoded with 128 bits, k is set to 100, and the Recall@i performance is reported when i ranges from 1 to $10,000$. From the figure, we can see that the Recall@i of *Auto-JacoBin* is consistently better than the performance of other hashing methods. DAutoBin, CAutoBin and the AutoBin1 have comparable results, and AutoBin2 gives the lowest Recall@i. Although the objective function of AutoBin1 and

AutoBin2 is the same, their performance is quite different. The performance of AutoBin1 is comparable to the performance of DAutoBin, while the performance of AutoBin2 gives the lowest Recall@i. From the comparison we find that the mini-batch stochastic gradient decent optimisation has the advantage of finding better local minima.

From our experiments, we note that DAutoBin is slightly better than AutoBin1, and AutoBin1 is slightly better than CAutoBin. The α in CAutoBin is set to be 0.01. We found that the performance of CAutoBin decreases when α increases in our experiments. The possible reason is that the denoising effect in CAutoBin is achieved implicitly by balancing two components. Due to the contractive component, the recovery of the original data points is not as good as using auto-encoder directly. Thus the performance of CAutoBin is decreased slightly. For DAutoBin, the noise is injected manually and the neural network is trained to recover the data point. The intuition behind this model makes sense since it aims to remove noise automatically. One of the problems for this approach is that it is unclear why the original data is the best recovery of the noisy data. Here is an extreme case, suppose x_1 and x_2 are two close but different data points, thus they might have the same noisy data point $\overline{x}$. According to the definition, when $\overline{x}$ is an input data point, the output of DAutoBin is assumed to be x_1 and x_2 at the same time. This contradicts the fact that x_1 and x_2 are two distinct points. On the other hand, Auto-JacoBin does not have this kind of contradictory behaviour since it estimates the best projection of the noisy data point. We believe this is the main reason that Auto-JacoBin has a better performance than DAutoBin.

6.4.3 Results on benchmark data sets

The computational task is to find the k nearest neighbour points in the base data set for each query. Here we evaluate scenarios where $k \in \{1, 5, 10, 50, 100\}$. The Recall@$i$ performance is reported when i ranges from 1 to $10,000$. Therefore, only a part of the base data set is retrieved (1%, 0.1% and 0.1% for the NUS-WIDE, GIST1M and SIFT1M respectively). For both NUS-WIDE and SIFT1M, we report the Recall@i performance when the number of bits, which are used to encode the data points, is up to its feature's dimension, i.e., they are tested with $64, 96$ and 128 bit encodings. For the GIST1M data set, more bits are used to encode the data points because the underlying features have more dimensions. In our experiments, GIST1M is tested with $64, 128$ and 256 bit encodings. The results for NUS-WIDE are shown in Fig. 6.7 and the results for GIST1M is in Fig. 6.8. The Recall@i performance of different hashing approaches for SIFT1M, as they are shown in Fig. 6.9, are

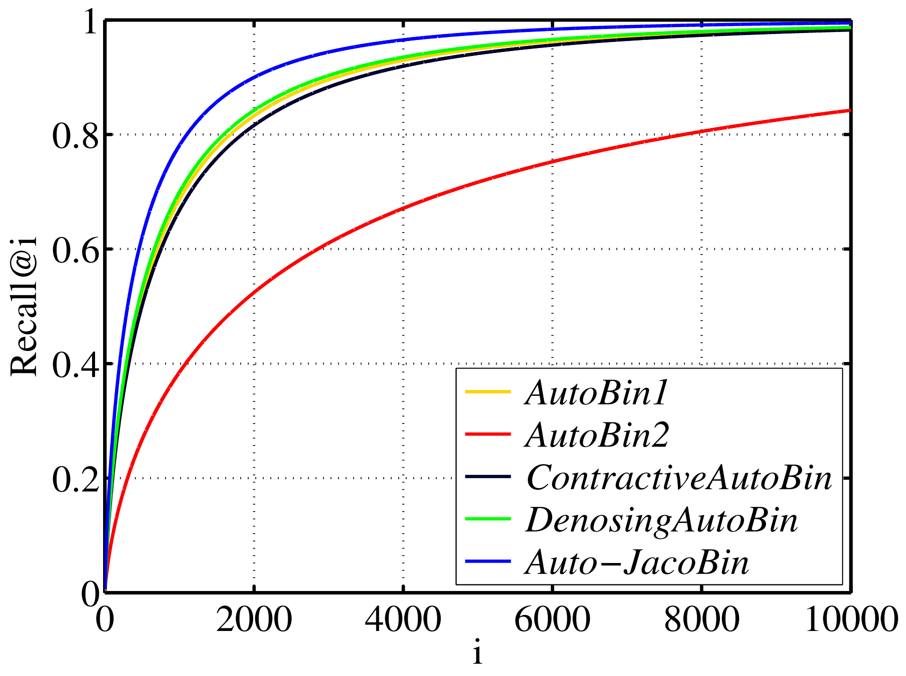


Fig. 6.6. Retrieval performance on the NUS-WIDE data set by different auto-encoder models used in the optimisation objective. Each data point is encoded with 128 bits, and the retrieval task is to find the 100 nearest neighbour points for each query.

much closer and more difficult to see in a graph, and therefore we present the m-Recall in a table. Table 6.2 shows the m-Recall performance of different hashing methods on the SIFT1M data set. Each row corresponds to one specific setting, where (i, j) means j bits are used for encoding the data set and the retrieval task is to find the i nearest neighbour points for each query.

In all the three figures (Fig. 6.7–6.9), the first row is the Recall@i performance for the nearest neighbour data point search task when the data points are encoded with different number of bits. The SIFT1M and NUS-WIDE are encoded with $64, 96, 128$ bits respectively and the GIST1M is encoded with $64, 128, 256$ bits respectively. The three pictures display the Recall@i performance when different number of bits, in increasing order, is used to encode each data point. The following rows show the comparison results for different retrieving tasks ($5, 10, 50$ and 100 nearest neighbour searching respectively) when the corresponding bits are used to encode the data points.

From Fig. 6.7 and Fig. 6.8, it is clear that the proposed method performs better than previous techniques. Other observations are that (unsurprisingly) the utilisation of more

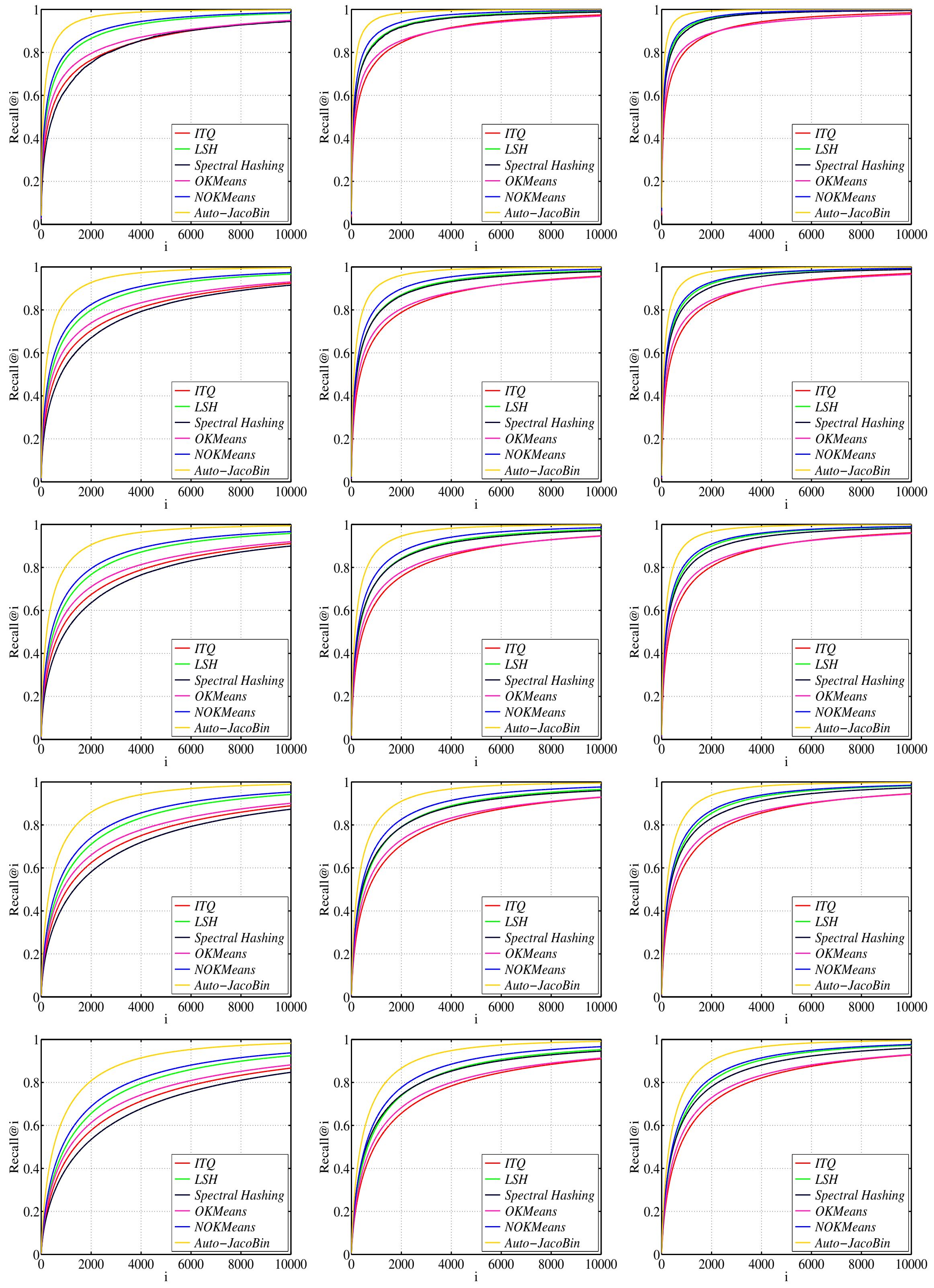


Fig. 6.7. Retrieval performance on the NUS-WIDE data set. The rows from top to bottom show retrieval performance for $k = \{1, 5, 10, 50, 100\}$ (nearest neighbours). The columns from left to right show the performance with $64, 96, 128$ encoding bits respectively.

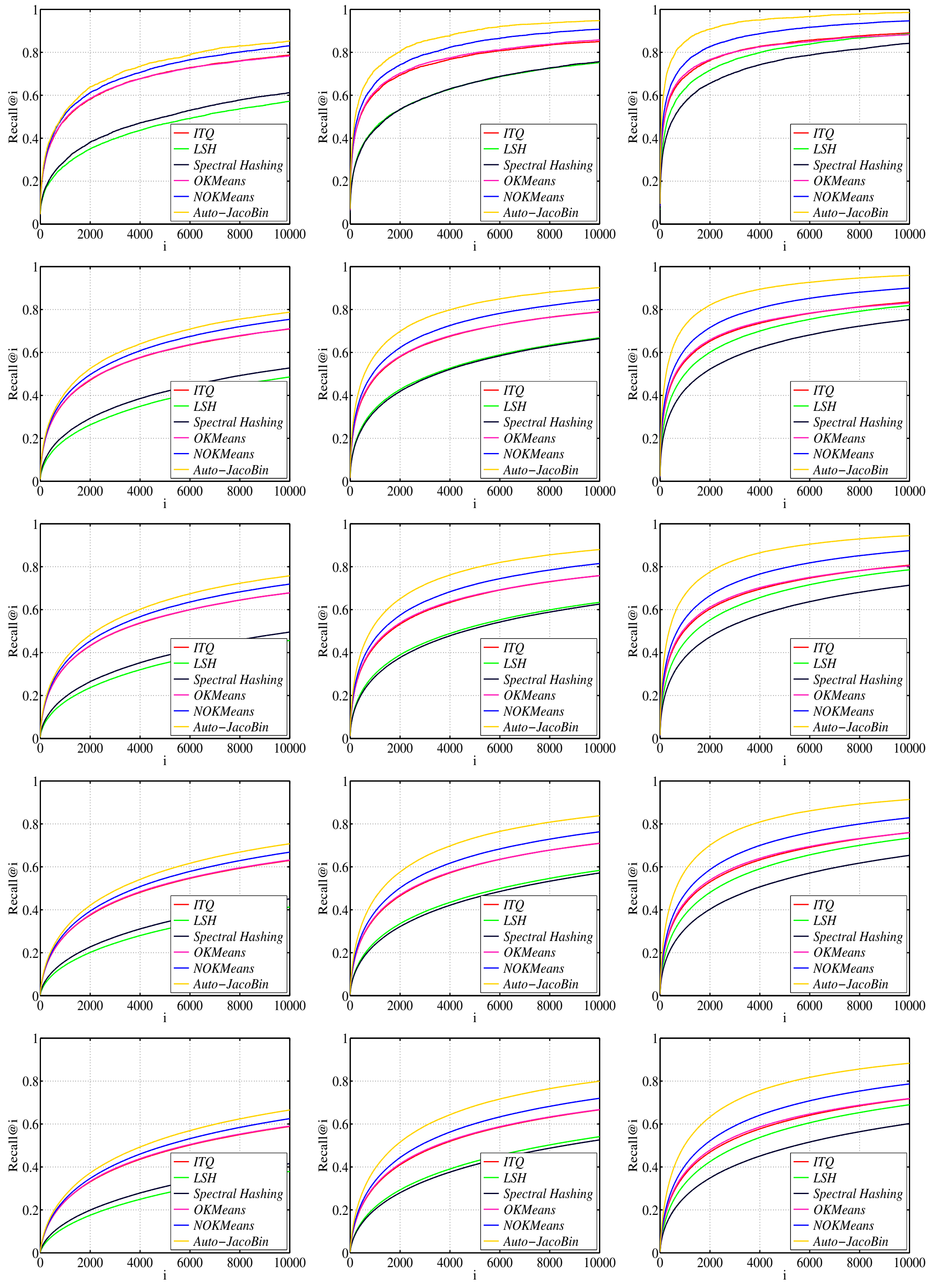


Fig. 6.8. Retrieval performance on the GIST1M data set. The rows from top to bottom show retrieval performance for $k = \{1, 5, 10, 50, 100\}$ (nearest neighbours). The columns from left to right show the performance with $64, 128, 256$ encoding bits respectively.

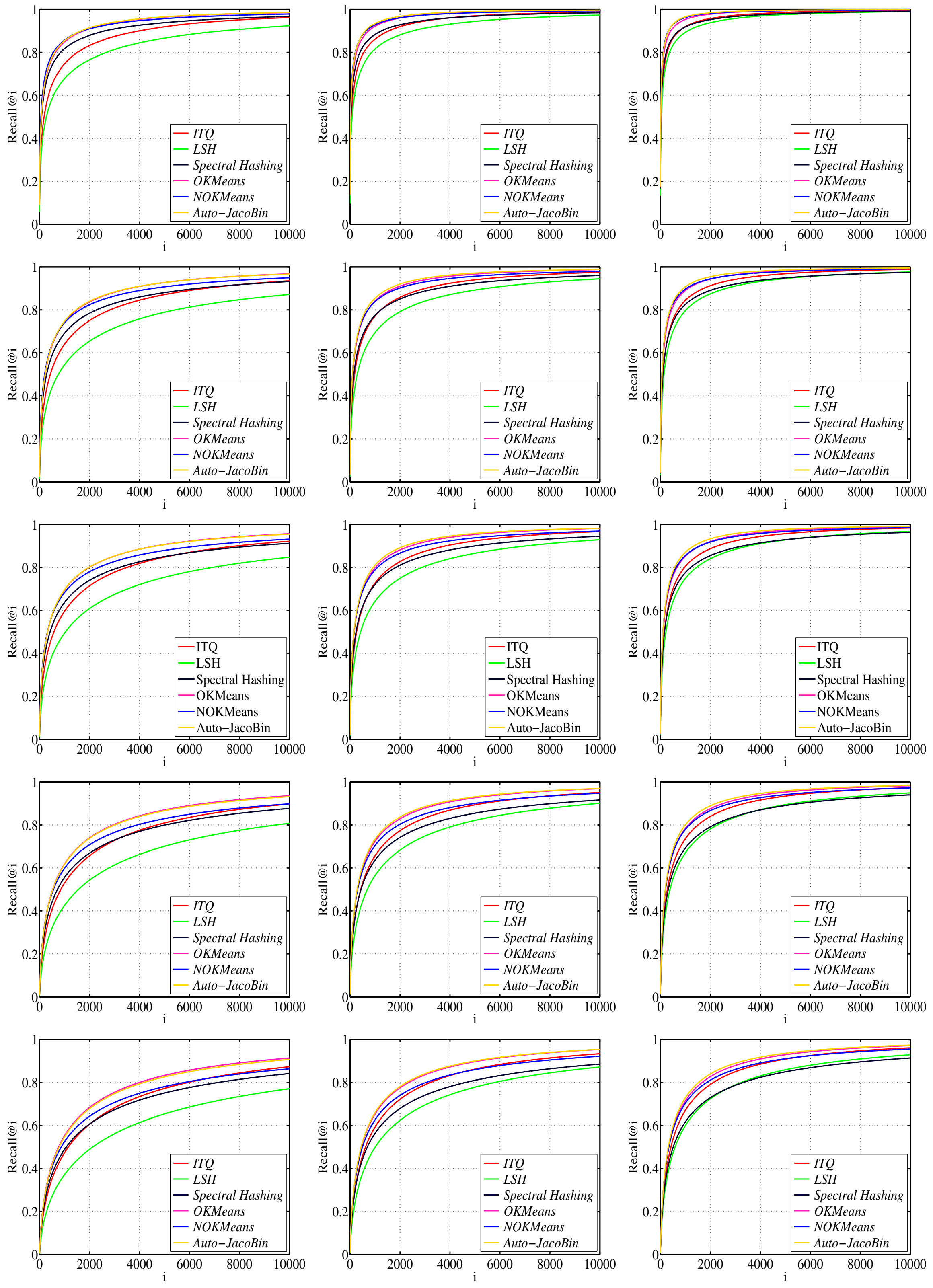


Fig. 6.9. Retrieval performance on the SIFT1M data set. The rows from top to bottom show retrieval performance for $k = \{1, 5, 10, 50, 100\}$ (nearest neighbours). The columns from left to right show the performance with $64, 96, 128$ encoding bits respectively.

bits results in better performance for all methods, and the performance decreases as k increases. Somewhat surprisingly, the data independent *LSH* algorithm performs comparably with many of the data-dependent algorithms, especially as the number of bits increases. One of the main reasons is that when modelling the data distribution, the data dependent hashing algorithms are done by some kind of relaxation such as obtaining the binary code from some threshold or encoding the data point into its nearest binary code, it is inevitable that the information contained in the original data set is lost to some degree when a limited number of bits are used to encode each data point during this process. While for *LSH*, the obtained binary codes can preserve the similarity between the data points albeit the projection is randomly generated. Another interesting point is that the comparative performance of different hashing algorithms depends on the data set. For NUS-WIDE, when few bits are used to encode the data points, *Spectral Hashing* has the lowest score. When the number of bits is increased, the performance of *Spectral Hashing* is comparable to *LSH* and *NOKMeans*, and has much better performance than *ITQ* and *OKMeans*. Both *ITQ* and *OKMeans* assume that their projection matrix is orthogonal. This constraint limits the separating capacity of different bits. That is why when more bits are used for encoding of the data points, the performance does not increase as much as other algorithms. For the proposed hashing method and *Spectral Hashing* and *LSH*, the projection matrices do not have this kind of constraint. Thus, the potential partition capability of these algorithms is better.

Overall, the benefits of the proposed method are clear. Even with a limited bit budget, the proposed hashing method has excellent Recall@i performance. For instance, when 64 bits are used to encode each data point on the NUS-WIDE data set, Recall@1000 is 0.92, which is 13% higher that state of the art results (the previous best is 0.81 from the *NOKMeans*). Also, to ensure good recall, when using 256 bits on the GIST1M data set, the proposed method requires $K = 2,000$ to achieve a 0.9 Recall@i. Compared with $K = 6,000+$ for the other methods, ours allows for improved computational performance in practice.

For the local feature data set, the proposed method has comparable performance to state of the art results (*NOKMeans*, *OKMeans* and *ITQ*). From Fig. 6.9, we can see that the yellow curve is always in the top position albeit it is overlapped with other leading performance. For better visualisation, we summarise some of the results based on the m-Recall indicator. Table 6.2 shows the m-Recall when the data points are encoded with different bits and for different retrieving tasks. From this table, we can see that the proposed method has comparable performance with other state of the art algorithms. Its performance becomes

Algorithms	*ITQ*	*LSH*	*Spectral Hashing*	*OKMeans*	*NOKMeans*	*Auto-JacoBin*
(1, 64)	0.879	0.825	0.908	0.933	0.930	**0.935**
(1, 96)	0.940	0.912	0.944	0.967	0.967	**0.971**
(1, 128)	0.965	0.952	0.962	0.979	0.982	**0.983**
(5, 64)	0.842	0.772	0.865	0.903	0.893	**0.905**
(5, 96)	0.915	0.872	0.910	0.947	0.943	**0.952**
(5, 128)	0.946	0.926	0.936	0.965	0.966	**0.971**
(10, 64)	0.820	0.740	0.837	0.882	0.867	**0.883**
(10, 96)	0.897	0.847	0.887	0.931	0.924	**0.937**
(10, 128)	0.933	0.907	0.916	0.953	0.953	**0.960**
(50, 64)	0.752	0.650	0.749	**0.814**	0.779	0.811
(50, 96)	0.839	0.767	0.806	0.876	0.853	**0.881**
(50, 128)	0.884	0.842	0.844	0.908	0.898	**0.915**
(100, 64)	0.714	0.604	0.699	**0.773**	0.728	0.767
(100, 96)	0.803	0.722	0.757	0.840	0.807	**0.843**
(100, 128)	0.852	0.802	0.797	0.876	0.859	**0.883**

Tab. 6.2. Retrieval performance (m-Recall) of different hashing methods on the SIFT1M. Each row corresponds to one specific setting, where (i, j) means j bits are used for encoding the data set and the retrieval task is to the find i nearest neighbour points for each query. For these tests, $K = 10,000$.

better when more bits are used to encode the data points. Another observation we can make is that *LSH* is consistently the worst, or nearly the worst, performer for this data set.

6.5 Summary

In this work, we have proposed a novel hashing algorithm which adopts the auto-encoder model to produce binary codes that are geometrically consistent with the data points in the original space. We have introduced a new auto-encoder objective function that makes use of the first order properties of geometric conservation (the Jacobian matrix) and has good noise-reduction properties. We proved that the Jacobian of the defined function has an intimate relationship with the tangent space of the manifold of the training data set, and show how the Jacobian can be expressed analytically.

The experiments are conducted on three large scale feature data sets. The performance of the proposed method is compared with several state of the art hashing algorithms. It has the best performance for global image features and comparable performance for the SIFT feature data set.

Chapter 7

How to select hashing bits? A direct measurement approach

Note: Some portions of this chapter are taken from Fu *et al.* (2015).

In the previous two chapters, we have discussed two methods to learn encoding functions. These two hashing methods are based on some geometrical assumptions and learn the encoding functions directly by solving corresponding optimisation problems. Since different hashing methods have different geometrical assumptions, the encoding functions generated from different hashing methods might complement one another, i.e., using a set of encoding functions which are generated from different hashing methods might lead to better retrieval performance. In this chapter, we address the problem of selecting an optimal set of bits (of a certain size) from a larger bit pool which is generated from any previous hashing method. We propose to assign a set of encoding functions with a score, which is calculated with information from data points in the local region and has an intimate relationship with the retrieval performance. In this way, the optimisation objective is to find a set of encoding functions with the highest score.

7.1 Introduction

In practice, generating candidate encoding functions (bits) for hashing problems is relatively easy. For example, one could use random hyperplanes (*LSH*) in the original space, or any of the previous hashing methods proposed for generating bits. More recently, focus has shifted from generating the best bits, to selecting a few bits from a large bit pool (Liu, He, Lang, and Chang, 2013). Such bit selection methods typically select bits which are maximally independent and best preserve similarity between data points. For example, one might try to maximise the correlation between the Hamming distance in hash space and the Euclidean distance in the original space. Although very sensible, preserving such similarities is only a proxy for the true objective – retrieving the true nearest neighbours.

In this work, we propose to measure the quality of a set of bits using retrieval performance directly. Thus, the optimal bit set is the one with the highest score. Measuring all of the possible subsets is impractical due to a combinatorial explosion. For example, selecting 64 bits from a pool with $1,000$ bits, we have to measure the quality of $C(1000, 64) \approx 10^{102}$ different bit sets. In view of this, we propose an alternating greedy optimisation method to find a local maximum. Actually, the main purpose is to find a set of bits such that it can be used to retrieve the true nearest neighbour points effectively. Therefore, there do exist many reasonable solutions, which are good enough for the retrieval task, and this is the main reason that our optimisation is efficient and effective.

Our main contributions are as follows:

- We investigate the approximate nearest neighbour search problem in large scale data sets, and then propose to measure the quality of a bit set by m-Recall which is appropriate for this retrieval task.
- An alternating greedy optimisation method is proposed for the combinatorial optimisation problem. The experimental results show that it is effective and efficient for the bit selection problem.

7.2 Background

7.2.1 Notation

Suppose $X = [x_1, x_2, \cdots, x_N] \in \mathbb{R}^{D\times N}$ is the training data set. The bit pool $\mathcal{P}$ consists of L bits which can be generated by any previous hashing algorithms. Denote $B \in \{-1, 1\}^{N\times L}$ as the binary matrix of the training data set encoded by the pool. The task of bit selection is to choose $\mathcal{S} = \{b_1, b_2, \cdots, b_M\}$ with M bits from the pool $\mathcal{P}$. Denote k as the number of true nearest neighbour points for each query and K as the maximum number of data points retrieved in Hamming space.

7.2.2 Related work

With the hashing methods proposed previously, many encoding functions are available to encode data points. Due to storage and retrieval efficiency considerations, we want

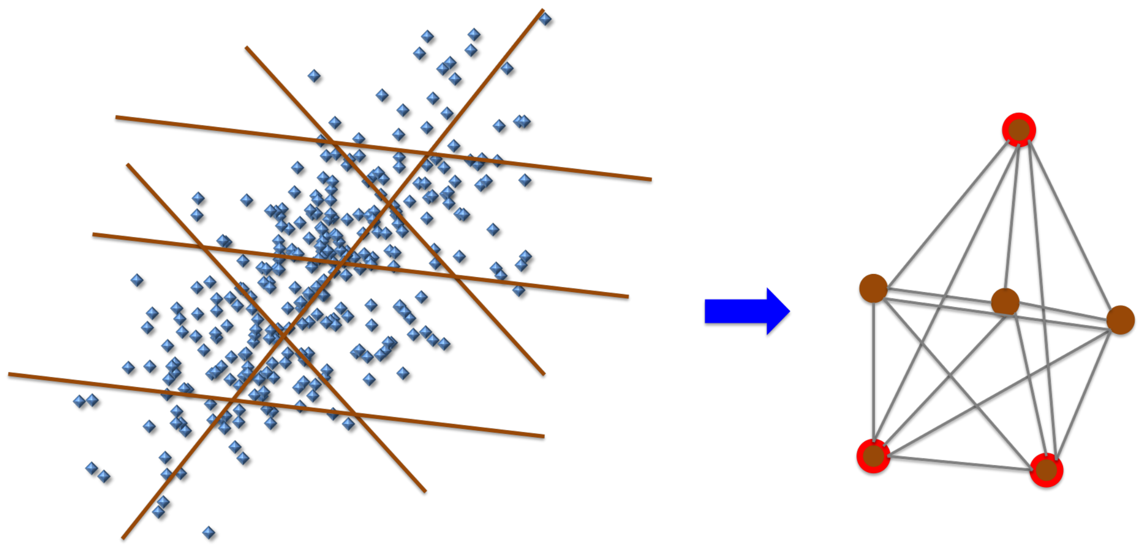


Fig. 7.1. Visualisation of a previous bit selection method. In the left picture, each line is an encoding bit from the pool. The pool is represented as a graph which is obtained by the similarity preservation information of the bits and the mutual information between the bits. The three red nodes, which correspond to the solution of the optimisation problem, are selected as the final encoding functions.

to encode the data points with few bits. Rather than limit the pool to a single method of generating bits, one could hope that different hashing algorithms might complement each other, and thus lead to a richer bit pool from which to select bits and ultimately a better encoding scheme. In a previous bit selection method (Liu *et al.*, 2013), two kinds of information are used in order to select a set of bits from the pool. First the selected bits should capture the local geometric information of the original data set. The second criterion is that the mutual independence information between the selected bits should be minimised. The bit selection problem is described as a dense subgraph discovery problem, and the final optimisation problem is formulated as:

$$\begin{aligned} \arg\max_{x} \quad & \frac{1}{2} x' A x \\ s.t. \quad & x \in \{0,1\}^L \\ & |x|_0 = M \end{aligned} \tag{7.1}$$

where A incorporates both the similarity preservation and the mutual independence information of the bits from the pool. The support of the solution x corresponds to the bits to be selected. This approach is typical of much previous work on bit generation and bit selection where the goal is to produce a set of independent bits that maintain the metric structure of the original data as much as possible. Such approaches are very suitable for dimensionality reduction and binary embedding problems but, although sufficient, are not necessary for retrieval tasks.

7.3 Algorithm

7.3.1 Motivation

Much previous work on binary hashing for ANN focuses on preserving similarity or order information in a local neighbourhood (Datar *et al.*, 2004; He *et al.*, 2013; Indyk and Motwani, 1998). In ANN however, especially for image retrieval applications, the goal is to retrieve the most relevant data-points and rank them highly among all results retrieved. Preserving similarity is a sufficient condition for achieving this goal. Alternatively, preserving order information (Wang, Wang, Yu, and Li, 2013) is also a sufficient condition. However, when retrieving multiple data-points, maintaining the order may also not be necessary, depending on the application. These observations motivate us to seek a direct way to select bits from a bit pool rather than through maintaining similarity or order information extracted from the original data set.

As discussed in section 5.4.1, m-Recall is a suitable indicator for measuring the performance of hashing methods for ANN tasks. In practice, it is infeasible to calculate m-Recall directly for each set of encoding functions since the data set is of large scale and high dimension. It would take too much computation to encode the whole data set, generate the query data points, and then calculate the m-Recall. Indeed if this were feasible it would be feasible to calculate the nearest neighbours directly. The training data set is assumed to have the same distribution as the base data set but a much smaller size which enables us to learn the encoding functions. As opposed to common hashing methods which aim to extract information from the training data set and then learn the encoding functions, we propose to calculate the m-Recall of a set of encoding functions on the training data set. Specifically, we only need to set two parameters: the number of true nearest neighbour points k_G and the maximum number of data points to be retrieved K_G for the training data set in order to mimic the ANN task in base data set.

7.3.2 Optimization

The task of bit selection from a pool is to find a fixed size subset such that the cost according to some measurement is as high as possible:

$$\begin{aligned} \arg\max_{\mathcal{S}} \quad & \text{m-Recall}(\mathcal{S}) \\ s.t. \quad & \mathcal{S} \subset \mathcal{P} \\ & |\mathcal{S}| = M \end{aligned} \tag{7.2}$$

This is a combinatorial optimisation problem, and it is NP hard (Hamo and Markovitch, 2005). We propose to obtain a reasonable solution by an alternating greedy (*AGreedy*) optimisation method, which is summarised in Algorithm 8.

Algorithm 8 Bits selection with alternating greedy method

```
    Input: Training data points $x_1, x_2, \cdots, x_N$; bit pool $\mathcal{P}$; maximum number of iterations $I_{max}$; size of the selected bit set $M$.
    Output: The selected bit set $\mathcal{S}$.
 1: Initialize $\mathcal{S}$ with $M$ bits which are randomly sampled from $\mathcal{P}$. Denote the m-Recall of the sampled $M$ bits on the training data set as m-Recall($\mathcal{S}$).
 2: for $i = 1$ to $I_{max}$ do
 3:     for $j = 1$ to $M$ do
 4:         Prepare the candidate bit set $\mathcal{C} = \mathcal{P} - \mathcal{S}$.
 5:         Precompute Hamming distances $\mathbf{D}_{\text{Pre}} \in \mathbb{R}^{N \times N}$ based on the $M - 1$ bits from $\mathcal{S} - b_j$. Here $b_j$ is the $j^{\text{th}}$ bit in $\mathcal{S}$.
 6:         for $k = 1$ to $|\mathcal{C}|$ do
 7:             Update the Hamming distance $\mathbf{D}_k$ based on $\mathbf{D}_{\text{Pre}}$ and the bit $b_k$.
 8:             Compute the m-Recall $m_k$ based on $\mathbf{D}_k$ and the true nearest neighbour information.
 9:         end for
10:         Find $b_{\overline{k}}$ such that $m_{\overline{k}}$ is the largest among $m_1, m_2, \cdots, m_{|\mathcal{C}|}$.
11:         if $m_{\overline{k}} >$ m-Recall($\mathcal{S}$) then
12:             Update m-Recall($\mathcal{S}$) with $m_{\overline{k}}$.
13:             Update the $j^{\text{th}}$ bit in $\mathcal{S}$ with $b_{\overline{k}}$.
14:         end if
15:     end for
16: end for
```

When a pool is given, the task of bit selection is to find M bits from the pool. In order to measure the quality of the selected bits on the training data set, we test its performance when part of the training data set is retrieved, and calculate the m-Recall as a measurement of the quality of the selected bits (Line 8). Each bit position is updated sequentially – that is, first bit 0 is updated, then bit 1, etc. The best candidate bit from the bit pool is chosen (Line 11–14) by evaluating all bits left in the pool (Line 6–9). This updating approach has two computational benefits. First, for each candidate bit the computation is independent and therefore it is easy to parallelise. Second, since each bit position is updated sequentially, the remaining $M - 1$ bits are fixed. Therefore, the Hamming distance for those $M - 1$ bits can be precomputed (Line 5), with the final Hamming distance computed as the sum of the precomputed distance and the distance of the candidate bit (either 0 or 1 – Line 7).

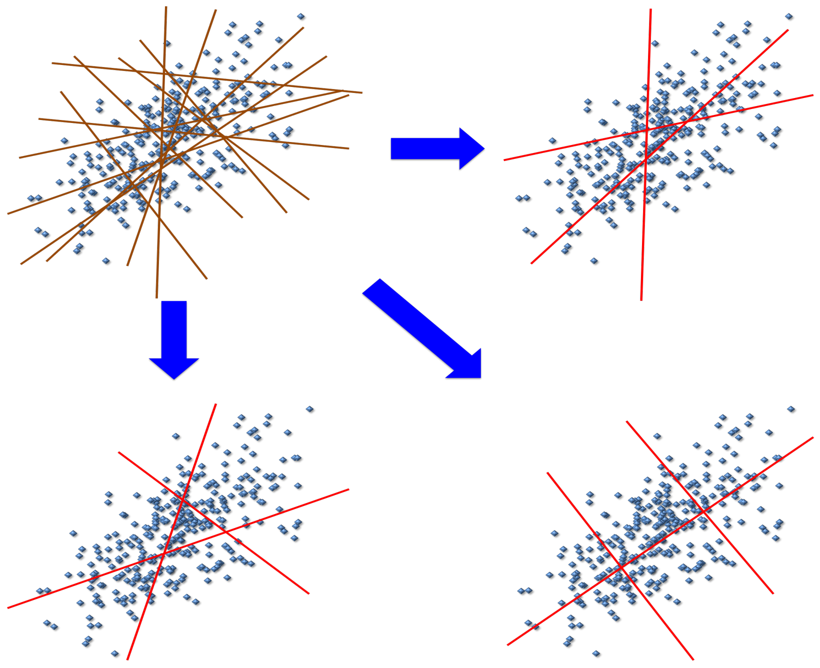


Fig. 7.2. Visualisation of multiple suboptimal solutions. The top left figure shows the bit pool. The remaining three figures show three bits selected from the bit pool. Although the solutions are totally different, the retrieval performance is similar.

Although the proposed alternating greedy (*AGreedy*) optimisation method only guarantees to find a local maximum, we believe it is effective to find a reasonable solution. One reason is that this optimisation problem has multiple local maxima, and it is relatively easy to find a reasonable solution by random initialisation. This is visualised in Fig. 7.2. The top left picture shows the pool of the encoding bits, and the other three pictures show the selected 3 encoding bits. Although the selection results are different, the performance of the selected bits have similar near optimal performance.

7.3.3 Computational complexity

The bulk of the computation is done by calculating m-Recall for each selected bit set. For each m-Recall calculation, it takes $\mathcal{O}(N^2)$ time to compute the Hamming distances between all of the training data points. Since the Hamming distances are discrete values which range from 0 to M, it takes $\mathcal{O}(N)$ time to find the K nearest neighbour points for each query point. Thus, for N queries, K nearest neighbour searching in Hamming space takes $\mathcal{O}(N^2)$. Finally, for updating each bit, we have to take $\mathcal{O}(|P|) = \mathcal{O}(L - M) = \mathcal{O}(L)$ time for m-Recall calculations, thus with I_{max} iterations updating S, it takes $\mathcal{O}(I_{max}MLN^2)$ in total.

7.4 Experiments

For evaluating the proposed bit selection method, we test its performance on three real large scale data sets: SIFT1M (Jégou *et al.*, 2011), GIST1M (Jégou *et al.*, 2011) and NUS-WIDE (Chua *et al.*, 2009). The feature vectors from these data sets cover both local and global image descriptors, and their dimension ranges from 128 to 960. The training set, query set and base set have the same setting as in the last chapter.

The performance of previous state of the art hashing algorithms is either included as baselines or used to generate the bit pools. *LSH* (Indyk and Motwani, 1998) is a data independent approach which generates encoding functions using a random Gaussian distribution. *Spectral Hashing* (Weiss *et al.*, 2008) is based on a model which inherits local geometric information from the original space. *ITQ* (Gong and Lazebnik, 2011), *OKMeans* (He *et al.*, 2013) and *NOKMeans* (Fu *et al.*, 2014) are based on minimising the quantisation error which is the gap between the original data points and the binary codes when both of them are viewed in Euclidean space. *NDomSet* (Liu *et al.*, 2013) uses a graph optimisation technique to select a bit set from a pool.

For each hashing algorithm, the binary encoding functions are learned based on the training data set only, and then used to encode the base data set and the query data set. For the bit selection methods, the pool is generated by a set of hashing algorithms. Only the selected bits from the pool are used to encode the base data set and the query data set. The performance of different hashing algorithms is compared based on the same settings including the number of bits, the number of true nearest points (k) to be retrieved and the number of data points to be retrieved (K) in the Hamming space.

For comparing the performance of different hashing methods, we use Recall@i and m-Recall as the indicators since they can be tailored to indicate the performance of the hashing methods in real ANN tasks. Hashing methods are mainly used for nearest neighbour searching tasks for large scale high dimensional data sets. The K and k are much smaller than the size of the base data set and this means ANN is not exactly the same as other information retrieval tasks. Although the optimisation objective in training stage is to find the bit set such that it has the highest m-Recall, we should note that the evaluations are taken on different data sets and the size of training data is much smaller when it is compared to the size of the base data set.

7.4.1 Parameter selection

For the proposed bit selection method, we only need to set two parameters. One is the number of data points to be retrieved (K_G) and the other is the number of true nearest neighbour points (k_G) for the training data set. The main purpose of this setting is to mimic the retrieval scenario in real large scale data. So, we empirically set K_G to be $1\% - 10\%$ of the training data set and k_G to be $1 - 200$ depending on K_G.

Fig. 7.3 shows the performance of the selected bits on the base data set. The performance is based on SIFT1M, the retrieval task is to find the nearest neighbour point for each query, and the corresponding m-Recall is reported for each set of parameters. The pool is generated by *LSH* with $1,000$ bits. The bit selection task is to choose 64 bits. From the figure we can see that K_G is relatively stable when it ranges from 5 to 200. Thus, the size of K_G does not affect the performance much. Based on this observation, we set K_G to 100 and k_G to 5 for all of the following experiments.

7.4.2 Comparing results with simulated annealing

When a subset is chosen, the cost is calculated by the m-Recall performance on the training data set. Thus, it is a combinatorial optimisation problem. Finding the global maximum and its corresponding subset is NP-hard. In practice, simulated annealing (Kirkpatrick, Gelatt, Vecchi, *et al.*, 1983) is widely used for this kind of non-convex discrete optimisation problem. The main idea of simulated annealing is to accept the candidate bit with some probability, which is related to the original cost, current cost as well as the stage of the overall iteration.

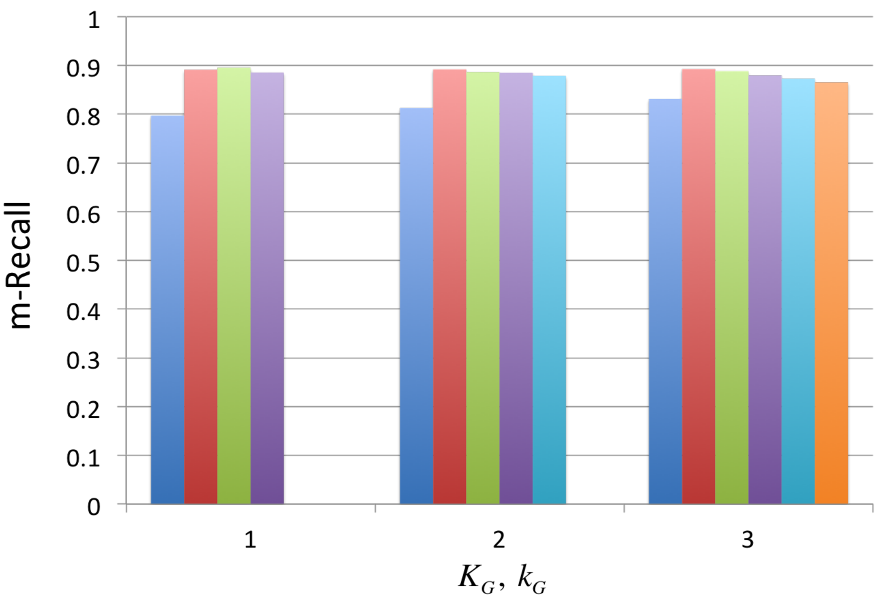


Fig. 7.3. Parameters with different retrieval settings. The pool is generated with $1,000$ bits. The maximum number K_G of retrieved data points is set to be $100, 500$ and $1,000$ respectively, and they correspond to the three groups in the figure. In the first group, k_G of true neighbourhood data points is set to be $1, 5, 10$ and 20. In the second group, k_G is chosen from $\{1, 5, 10, 50, 100\}$. In the third group, k_G is chosen from $\{1, 5, 10, 50, 100, 200\}$.

Comparing our proposed method with simulated annealing gives a good indication of how close the greedy method approaches a reasonable solution of simulated annealing.

For comparison with simulated annealing, the pool consists of $1,000$ bits which are generated by *LSH*. The bit selection task is to choose 32 bits which will be used for encoding the NUS-WIDE data set. Thus, in each iteration, there are $(1,000 - 32) = 968$ candidate bits. Initially bits which decrease the retrieval performance are accepted with probability 0.1, which decreases to 0.00001 over $1,936$ iterations. Bit selection with simulated annealing is similar to Algorithm 8. The only difference is that, in each bit iteration, simulated annealing accepts the randomly chosen candidate bit according to some probability; while the proposed optimisation method considers all of the candidate bits and then chooses the best one.

The related cost changes and the performance are compared in Fig. 7.4 and Fig. 7.5. Fig. 7.4 shows the cost changes of different optimisation methods. The left figure shows the cost changes of the alternating greedy optimisation method. The bit updating goes through two times, i.e., the bit set is updated 64 times. In each update, it takes $1,000 - 32 = 968$

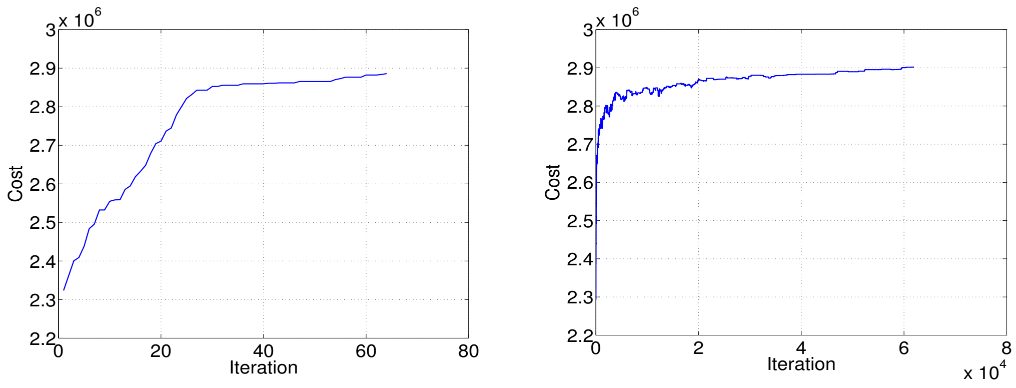


Fig. 7.4. The cost changes of different optimisation methods. The pool is generated by *LSH* with $1,000$ bits. The selection task is to choose 32 bits for encoding NUS-WIDE. The first figure shows the cost changes of the *AGreedy* method. The second figure shows the cost changes of the simulated annealing optimisation method. The total amount of m-Recall measurements is the same for both optimisation methods. The difference between them is that *AGreedy* takes 64 iterations and needs about 28 minutes to run in our 64-core machine, while simulated annealing takes $(32 \times 1,936)$ iterations and needs about 3 days to run.

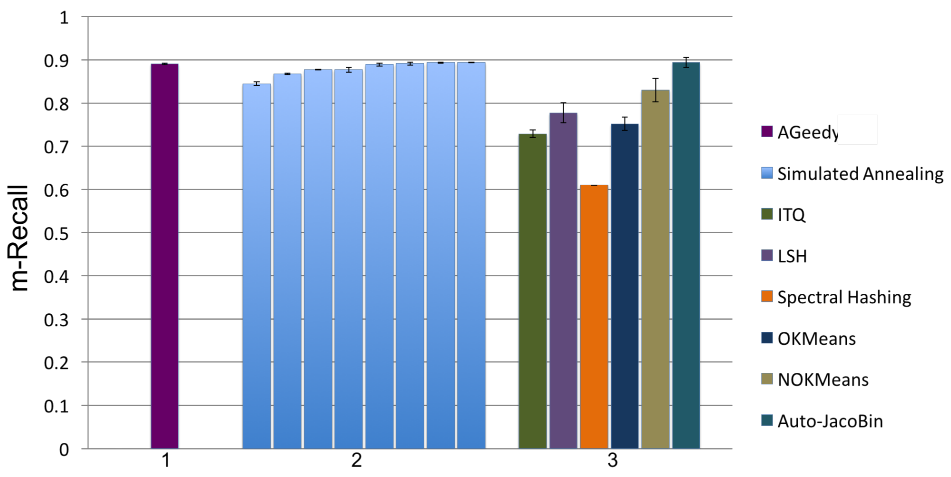


Fig. 7.5. Performance comparison with the simulated annealing optimisation method. The pool is generated with $1,000$ bits. The selection task is to choose 32 bits for encoding NUS-WIDE. The results are divided into three groups. The first group is the m-Recall performance of the proposed alternating greedy method. The second is the simulated annealing optimisation method. The experiment takes $(32 \times 1,936)$ iterations of the bit update. The m-Recall of the $(2 \times 32)^{\text{th}}$, $(10 \times 32)^{\text{th}}$, $(50 \times 32)^{\text{th}}$, $(100 \times 32)^{\text{th}}$, $(500 \times 32)^{\text{th}}$, $(1,000 \times 32)^{\text{th}}$, $(1,500 \times 32)^{\text{th}}$ and $(1,936 \times 32)^{\text{th}}$ iteration is reported. The third group summaries the corresponding results of other hashing methods used as baseline in the experiment. The standard deviations of the m-Recall are shown inthe corresponding bars.

times of m-Recall measurements. The second figure shows the cost changes of the simulated annealing optimisation method. The bit updating goes through $1,936$ times and the cost is updated $(1,936 \times 32)$ times. In each update, it takes one m-Recall measurement. From the left picture in Fig. 7.4, we can see that the first iteration of the bit updating of the proposed alternating greedy optimisation method is effective. The cost changes slightly in the second iteration. Since computation is the main concern during optimisation, we set a maximum of two iterations ($I_{max} = 2$) in all of the following experiments. Although there are only two iterations in *AGreedy*, it takes much more time to update each bit because it has to calculate m-Recall for each candidate bit. While in simulated annealing, each bit is updated with a randomly chosen bit according to some probability, i.e., only one m-Recall measure is needed to update the bit. To facilitate the performance comparison, we set the same amount of m-Recall measurements $(2 \times 32 \times 968)$ for both optimisation methods.

Fig. 7.5 shows the performance of the selected bits in a real large-scale data set. We repeat each experiment 5 times. The results are divided into three groups. The first group shows the performance of bits selected by the proposed alternating greedy optimisation method. The second group shows the performance of the bits selected by simulated annealing method. We take the results from the $(2 \times 32)^{\text{th}}$, $(10 \times 32)^{\text{th}}$, $(50 \times 32)^{\text{th}}$, $(100 \times 32)^{\text{th}}$, $(500 \times 32)^{\text{th}}$, $(1,000 \times 32)^{\text{th}}$, $(1,500 \times 32)^{\text{th}}$ and $(1,936 \times 32)^{\text{th}}$ iteration, and use them to encode NUS-WIDE respectively. With the null hypothesis that *AGreedy* and the method from i^{th} iteration have the same means of m-Recall but their variances might be different and the alternative hypothesis that *AGreedy* has a larger mean of m-Recall, the p-values for the first 4 tests are 1.00×10^{-6}, 1.74×10^{-8}, 1.36×10^{-5} and 1.50×10^{-3} respectively. According to the significance level $\alpha_t = 0.05$, *AGreedy* has significant better results than the methods from iterations up to (100×32). The third group shows the results from other baseline hashing methods, and they are ordered as *ITQ*, *LSH*, *Spectral Hashing*, *OKMeans*, *NOKMeans*, and *Auto-JacoBin*. Since the pool is generated by *LSH*, it is interesting see whether *AGreedy* and simulated annealing methods select better quality encoding functions. The p-values for the corresponding two-sample t-tests range from 1.20×10^{-3} to 1.61×10^{-4}. Thus, according to the significance level $\alpha_t = 0.05$, these methods select better quality encoding functions from the pool. Besides, the standard deviations of different hashing methods show that AGreedy has a relatively stable performance.

From Fig. 7.4 and Fig. 7.5, we can see that the proposed alternating greedy optimisation method has similar performance with the results from the simulated annealing method. The main advantage of the proposed method is that it can be run on a multi-core machine. Taking the experiments in this section as an example, the proposed method takes about 28 minutes for our 64-core machine. For the same amount of evaluations ($2 \times 32 \times 968$), the simulated annealing method takes about 3 days. From the time comparison, we can see that the proposed method has about 160 times speedup. This is due to the steps of parallel running and distance pre-computation as discussed in section 7.3.2. There are also some variants of the simulated annealing method which addressed the sequential implementation of simulated annealing (Onbaşoğlu and Özdamar, 2001). In view of the facts that *AGreedy* has comparable performance as the sequential simulated annealing method and can be implemented efficiently in multi-core machines, we use *AGreedy* for all of the following experiments.

7.4.3 Performance with different pools

For both *NOKMeans* and *NDomSet*, a set of choices is considered for each parameter, and then the best choice is fixed for all following experiments. For each data set, we show results in a Recall@i graph over all K for $k = 1, 5, 10, 50, 100$. For SIFT1M, we also report results for several values of k and K in tabular form, because the curves in an m-Recall graph are difficult to distinguish from each other.

We test the bit selection algorithms on different pool sources. Pool 1 is generated with $\{$*LSH*$\}$. Pool 2 is generated with $\{$*LSH*, *ITQ*, *OKMeans*, *NOKMeans*, *Auto-JacoBin*$\}$, and Pool 3 is generated with $\{$*LSH*, *ITQ*, *OKMeans*, *NOKMeans*, *Auto-JacoBin*, *Spectral Hashing*$\}$. When the pool size is fixed, each hashing method generates the same number of bits. The pool size for SIFT1M and NUSWIDE is 600, and the pool size for GIST1M is 1200. For different pools, the corresponding bit selection method will be distinguished by appending the pool index, i.e. *NDomSetP1*, *NDomSetP2*, *AGreedyP1*, *AGreedyP2* and *AGreedyP3*. The performance of *NDomSetP3* is not included since the corresponding performance is the worst among all of the baseline algorithms. The possible reason is that *Spectral Hashing* and *NDomSet* use similar optimisation functions, which affects the selection in *NDomSet*.

Fig. 7.6 displays the Recall@i performance of different hashing methods on NUSWIDE. The data set is encoded with $64, 96$ and 128 bits respectively, and for each query, the retrieval task is to find $1, 5, 10, 50$ and 100 nearest neighbour points. The performance of all algorithms tends to increase as the number of bit increase. The retrieval task becomes more challenging when more true nearest neighbour points are required to be retrieved. From the figure, we can see that *AGreedyP3* has the best performance. The possible reason is that it has the most variety since it includes random, quantisation error based and nonlinear

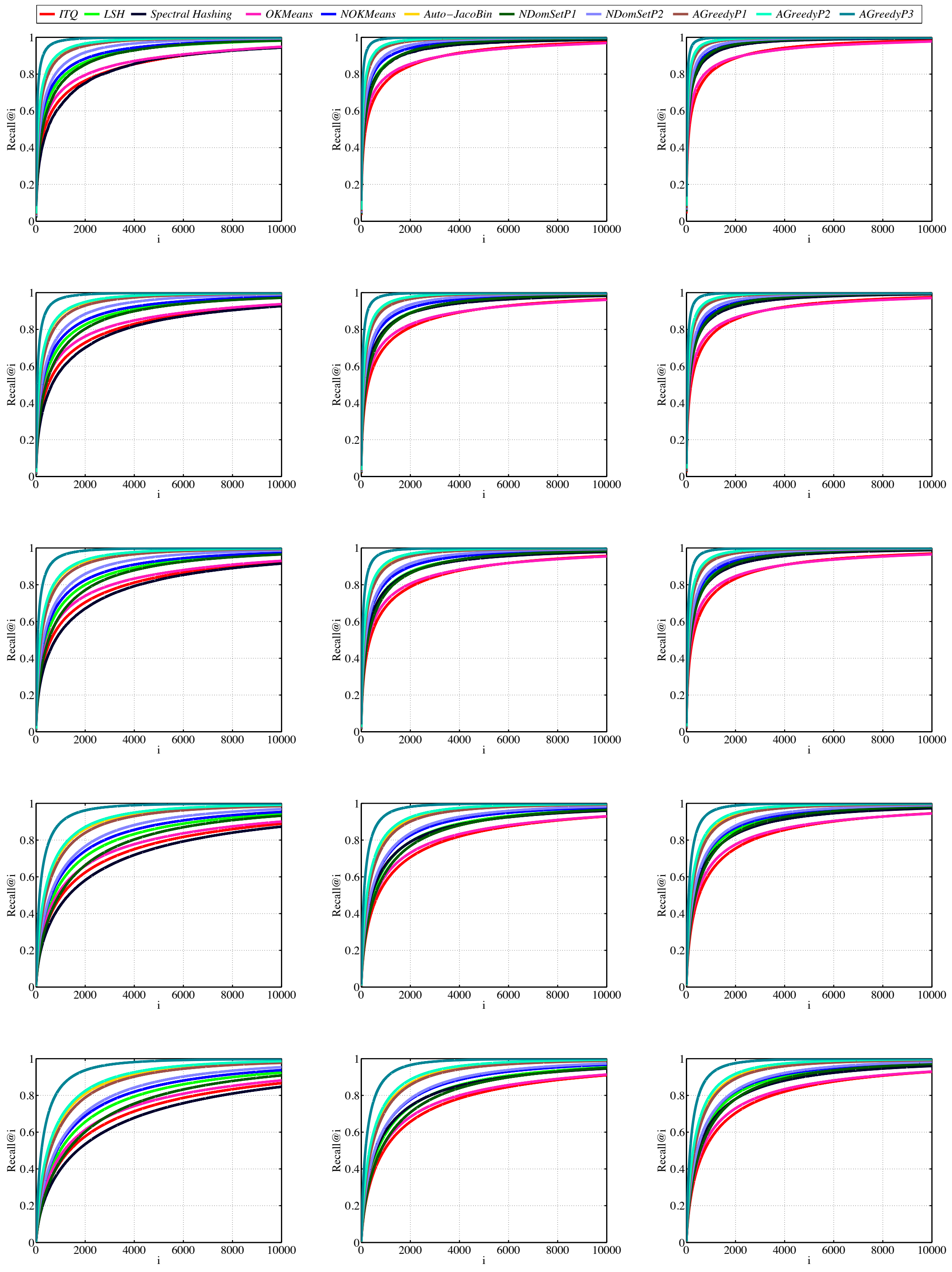


Fig. 7.6. Retrieval performance of different hashing methods on the NUS-WIDE data set. The first row shows the Recall performance for retrieving the nearest neighbour point for each query point. Each data point is encoded with $64, 96$ and 128 bits respectively. The following rows show the corresponding results for the $5, 10, 50$ and 100 nearest neighbour retrieving tasks.

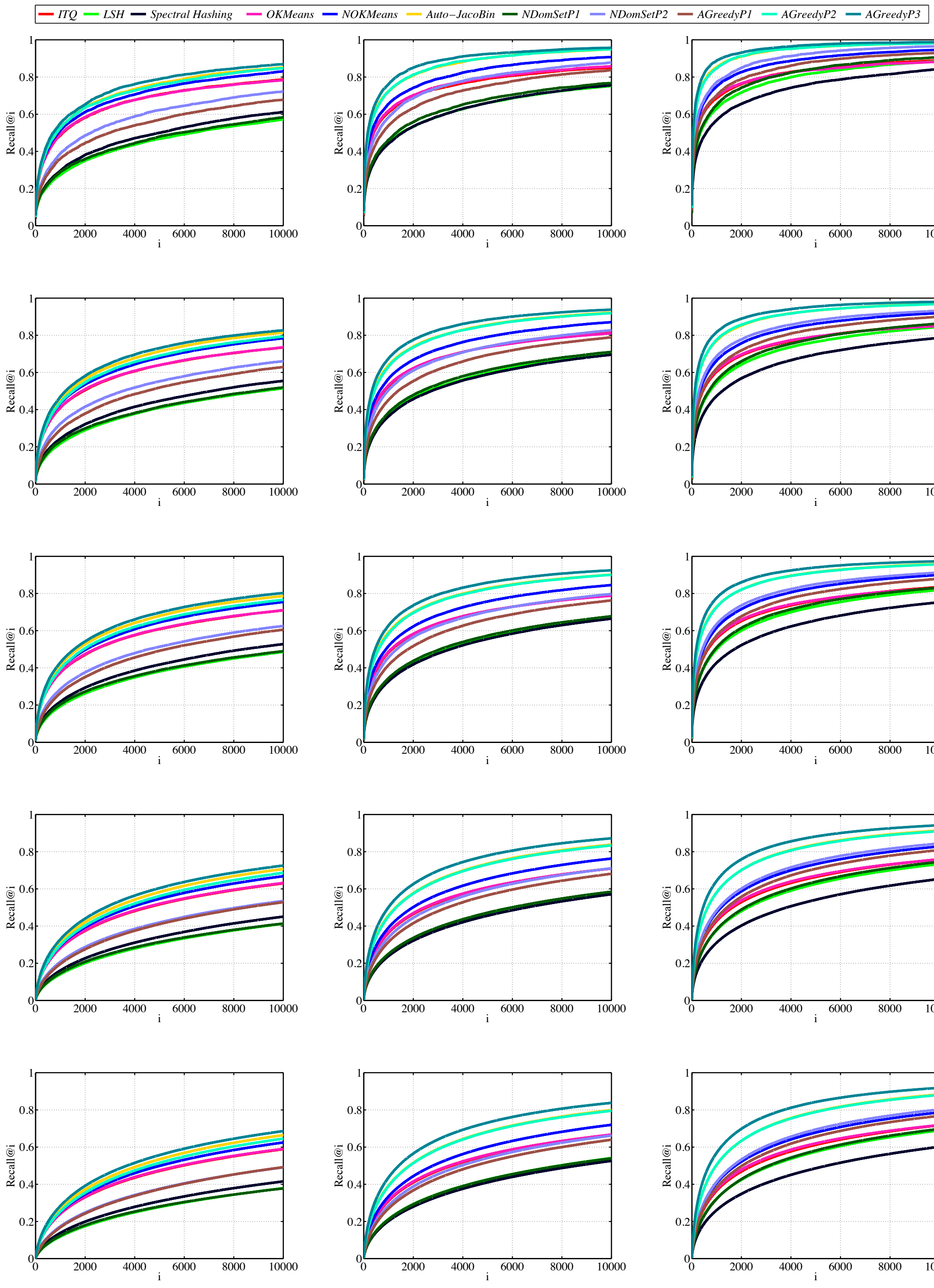


Fig. 7.7. Retrieval performance of different hashing methods on the GIST1M data set. The first row shows the Recall performance for retrieving the nearest neighbour point for each query point. Each data point is encoded with $64, 128$ and 256 bits respectively. The following rows show the corresponding results for the $5, 10, 50$ and 100 nearest neighbour retrieving tasks.

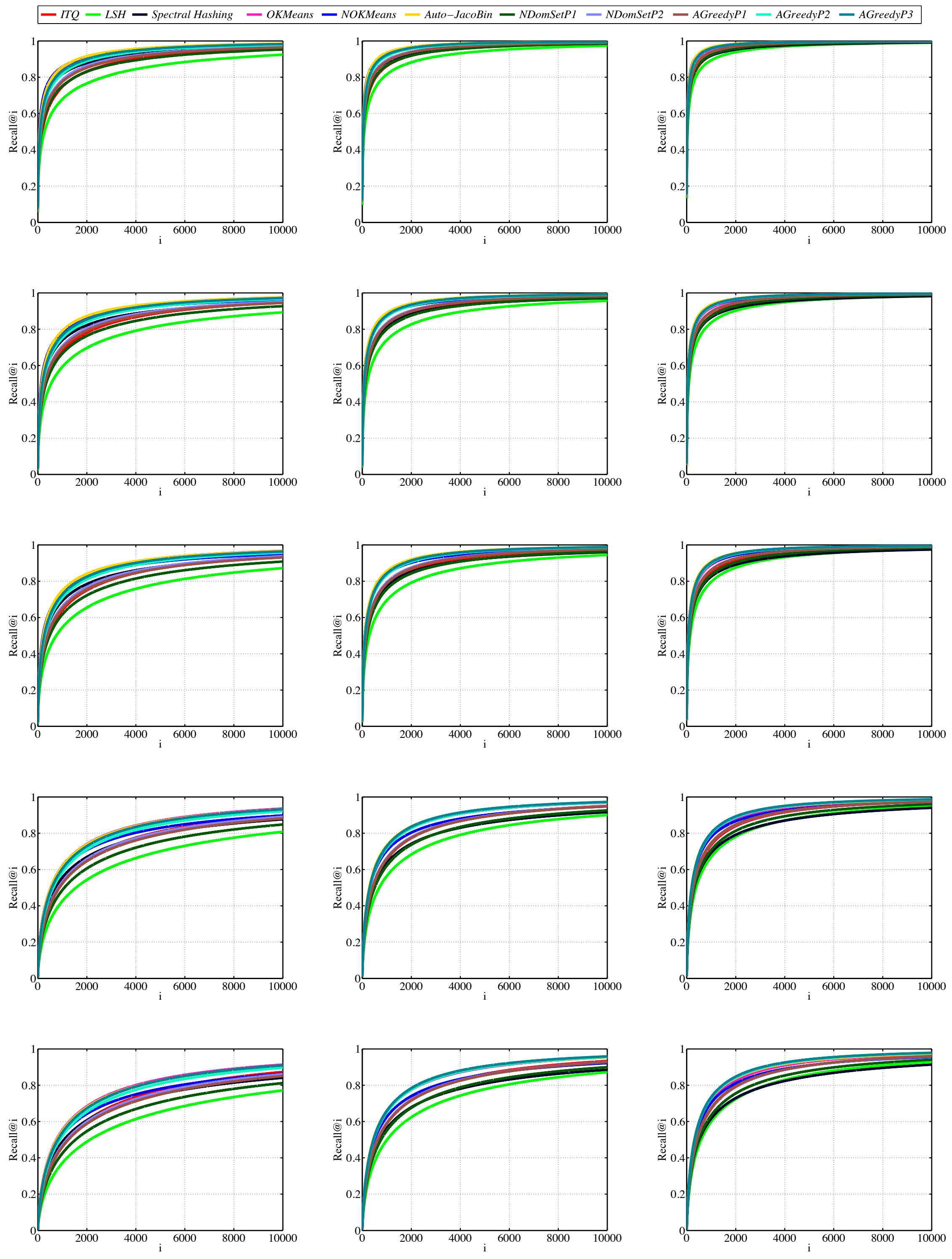


Fig. 7.8. Retrieval performance of different hashing methods on the SIFT1M data set. The first row shows the Recall performance for retrieving the nearest neighbour point for each query point. Each data point is encoded with $64, 96$ and 128 bits respectively. The following rows show the corresponding results for the $5, 10, 50$ and 100 nearest neighbour retrieving tasks.

encoding methods. *AGreedyP3* is much better than *AGreedyP2* which ranks second in most scenarios. The only difference between Pool 2 and Pool 3 are the *Spectral Hashing* bits, indicating that *Spectral Hashing* bits complement the bits from other hashing algorithms; despite the fact that *Spectral Hashing* alone is often the worse performer.

The retrieval performance for GIST1M has a similar trend as shown in Fig. 7.7. We can see that the bit selection algorithms have better performance when more hashing functions are used to generate the bits. *AGreedyP3* has the leading performance for GIST1M. *AGreedyP2* has comparable performance as *Auto-JacoBin* and this is the main reason that the yellow curve is invisible in most figures. It is interesting to compare the retrieval performance among *LSH*, *NDomSetP1* and *AGreedyP1* since Pool 1 is generated by *LSH*. We find that *NDomSetP1* does not always get much better performance than *LSH*. For example, when selecting 128 bits from Pool 1 on GIST1M, *NDomSet* has similar performance to *LSH*. For different k nearest neighbour retrieving tasks, the gap between *AGreedyP3* and the second leading performance is widened when k increases. Thus, *AGreedyP3* is capable for different retrieving scenarios.

Algorithms	*LSH*	*OKMeans*	*NOKMeans*	*Auto-JacoBin*	*NDomSetP1*	*NDomSetP2*	*AGreedyP1*	*AGreedyP2*	*AGreedyP3*
(1, 64)	0.825	0.933	0.930	**0.935**	0.873	0.898	0.892	0.917	0.926
(1, 96)	0.912	0.967	0.967	**0.971**	0.938	0.952	0.950	0.964	0.967
(1, 128)	0.952	0.979	0.982	**0.983**	0.966	0.974	0.973	0.981	0.981
(5, 64)	0.772	0.903	0.893	**0.905**	0.824	0.857	0.849	0.884	0.894
(5, 96)	0.872	0.947	0.943	**0.952**	0.906	0.926	0.923	0.944	0.948
(5, 128)	0.926	0.965	0.966	**0.971**	0.943	0.955	0.955	0.968	0.969
(10, 64)	0.740	0.882	0.867	**0.883**	0.793	0.831	0.822	0.863	0.873
(10, 96)	0.847	0.931	0.924	**0.937**	0.884	0.907	0.904	0.929	0.933
(10, 128)	0.907	0.953	0.953	**0.960**	0.926	0.941	0.941	0.957	0.959
(50, 64)	0.650	**0.814**	0.779	0.811	0.702	0.749	0.739	0.791	0.806
(50, 96)	0.767	0.876	0.853	0.881	0.808	0.841	0.838	0.876	**0.881**
(50, 128)	0.842	0.908	0.898	0.915	0.864	0.888	0.889	0.915	**0.918**
(100, 64)	0.604	**0.773**	0.728	0.768	0.654	0.704	0.694	0.750	0.766
(100, 96)	0.722	0.840	0.807	0.843	0.763	0.799	0.797	0.841	**0.847**
(100, 128)	0.802	0.876	0.859	0.883	0.824	0.851	0.854	0.886	**0.889**

Tab. 7.1. Retrieval performance (m-Recall) of different hashing methods on the SIFT1M. Each row corresponds to one specific setting, where (i, j) means j bits are used for encoding the data set and the retrieval task is to find the i nearest neighbour points for each query. For these tests, $K = 10,000$.

Fig. 7.8 and Table 7.1 shows the performance of different hashing algorithms on SIFT1M. Note that we have not presented the m-Recall for all of the hashing methods since some of them have been already summarised in Table 6.2. Although the difference between curves is difficult to distinguish, we can see that the curve for *AGreedyP3* is always among the top positions of the curves and the green line which corresponds to the performance of *LSH* is the lowest in most cases. From Table 7.1, we can see that *AGreedyP3* has comparable performance to other state of the art hashing methods and stands out when more bits are used to encode the data points. From the performance of *AGreedyP1*, we can see that *AGreedy* always selects better encoding functions since its performance is much better than *LSH*. We believe this is due to the direct measurement of the proposed bit selection method. The fact that the encoding bits from *LSH* are generated by random Gaussian distribution and $\mathcal{S}$ is randomly selected from Pool 1 ensures that the performance of *AGreedy* is comparable to *LSH* at the initialisation stage. For *AGreedy*, m-Recall improves with more iterations, thus its performance should almost always be better than *LSH*.

7.5 Summary

In this chapter, we focused on the problem of selecting bits from a bit pool. We used a direct measurement to evaluate the quality of a set of bits from the pool, and found a reasonable solution by an alternating greedy optimisation method. One might expect that a bit-selection algorithm that directly optimises the criteria of interest would do no worse than any of the bit-generation methods used to create the bit pool. This is true in two of the data sets tested here (NUSWIDE and GIST1M). For SIFT1M, this is not true when few bits are used, but *AGreedy* does better as more bits are added. With few bits it is more likely that a poor early bit-selection could not be overcome later.

Chapter 8

Conclusion

In this thesis, we have investigated the use of geometric information for three computer vision applications. We summarise the contributions according to how the geometric information is used:

- **Viewing covariance matrix of data points in the local region as Mahalanobis distance (Fu *et al.*, 2013).**

  For real-world clustering problems, such as the motion segmentation of video sequences, the implicit manifolds of different classes might be close to or intersect with each other. Data points near the intersecting regions might have large similarity with data points from other clusters. This is the main cause of failure in previous clustering methods. We propose to use the learned Mahalanobis distance to find the nearest neighbour data points. This allows us to learn a better similarity matrix such that the modified spectral clustering method has better clustering performance as measured by the misclassification rate. The experimental results on data sets, which are featured as the data points from different clusters intersect or are close to each other, show that the corresponding spectral clustering methods with the learned similarity have lower mis-classification rates.

- **Extracting subspace information for action classes and using this information for labelling query videos (Fu *et al.*, 2013).**

  In previous work, video sequences are represented as data points in some special manifold space (the product of Grassmannian manifolds). The nearest neighbour classifier is used for labelling the query video sequences. Due to the heavy computation required to calculate the distances between data points in the manifold space, the nearest neighbour classifier takes too much time to label the query video when the size of the training data set is large. We use the framework of principal geodesic analysis to learn the subspace information in a tangent space for each class. To label a new video sequence, we calculate its distance to the subspaces extracted in offline training stage. The class which corresponds to the subspace that has the smallest distance to the query video is predicted as the label. Thus, the computation for labelling a video sequence is proportional to the number of classes rather than the number of training points in training data set. The experimental results show that the proposed method takes less computation and has comparable recognition accuracy performance with previous related methods.

- **Representing data points in a new space (Fu *et al.*, 2014).**

  NOKMeans generalises the quantisation error based hashing methods which view the binary codes as special points in the original space and aim to minimise the gap between the original data points and the binary codes. The orthogonality of the hyperplanes is implicitly assumed in previous work. When the separating hyperplanes are orthogonal to each other, we have explicit indexing centres in the original space, and otherwise, it is impossible to obtain indexing centres in the original space due to the general position of the Voronoi diagram. In order to increase the representation capability of these binary codes, we relax the mutually orthogonal condition in quantisation error based hashing algorithms and address the existence of the indexing centres by viewing data points in a new space, which are determined by their distances to the hyperplanes. We index each data point by its nearest centre, the hyperplanes coincide with the boundaries of different index regions. This view benefits from the advantages of both quantisation error based hashing approaches and the non-orthogonal hyperplanes.

- **Mapping data points near the manifold space in the local region.**

  For the auto-encoder model, we assume that the noisy data point is distributed around the manifold, and its 'optimal' recovery is the nearest data point in the manifold. Although this function cannot be analytically expressed, it can be approximated by a first order form, which has an intimate relationship with the tangent spaces of the manifold. We use this auto-encoder model and a constraint on the hidden layer to obtain the binary codes. The auto-encoder model has the ability to keep the geometric information during forward propagation, and the constraint also has the effect of ensuring a better distribution of the binary codes in the sense that they are equally likely and uncorrelated between bits.

- **Selecting a bit set which preserves the order information in the local region (Fu *et al.*, 2015).**

  AGreedy selects a set of bits from a large bit pool. There are many hashing algorithms, but none of these algorithms are universally best for real data sets due to the variety of the high dimensional data sets. Each method is based on some geometrical model assumption which is used to approximate the geometric information of the original data set. Selecting some bits from a pool, which is generated from a rang of hashing methods, has the potential to find an hashing method which might better than any of the specific methods. We assign a score for each of the selected bit set. In this way, the bits selection is converted into a combinatorial optimisation problem, and we use an alternating greedy method to find a local optimal solution. This optimisation method is easy to implement on multicore computers and run in parallel. The use of shared data structures in this algorithm further reduces the amount of computation required.

We have tested the hashing methods on several large scale high dimensional data sets including both local and global image features. Experimental results show that the proposed methods achieve state of the art results on these data sets.

8.1 Future work

For the spectral clustering problem, we learn a similarity matrix from the Mahalanobis distance which is based on geometric information in the local region. The similarity matrix learning can be decomposed into two steps. The first step is to decide the connections between data points, and the second step is to assign a weight for each connection. In the region where data points from different classes is close to or intersect with each other, the learned Mahalanobis distance is used to find a better quality of the neighbourhood, i.e., the data points near the intersection region have more connections to the data points from the same cluster. Thus, the proposed method focuses on smart connections. In the future, we are interested in investigating how to calculate the similarity with the assistance of geometric information including the tangent space, parallel vector fields and the local curvature. Combining smart connections and smart similarity would have several benefits. For example, it would not only enable the data points to be connected to the data points from its own class, but also ensure that the similarity between them is strong. In some extreme cases, the data points might have mis-connections. The smart similarity is a possible remedy to this kind of problem since it aims to assign a small weight to such a connection.

For the action recognition problem, we have explored the use of subspace information for each action class. From our experiments, we can see that the geometric information can be used for labelling new action videos, and the computation time is proportional to the number of classes in training data set. In the future, we are interested in exploring more elaborate geometric information, such as the distribution of the data points in the product of Grassmannian manifolds space. It would be also interesting to investigate the hashing approach for the data set in manifold space by encoding the data points in manifold space into binary codes. Since the representation of the video sequences in manifold space is faithful and the hashing methods guarantee fast distance computation, labelling of new query videos might be further sped-up while maintaining the recognition accuracy.

For the hashing-based ANN tasks, we have investigated how to learn the encoding functions directly and how to select a set of encoding functions. Currently, we focus on a single view of the computer vision data set, since each image is represented by one vector such as the GIST feature. Thus, the main task is to find nearest neighbour points with same representations. In practice, one image might be described by multiple different descriptors, such as BOW or GIST features, or may be associated with text information. We would like to explore how to use multiple features and retrieving images when the query is given in form of textual information. As initial work in this direction, we plan to investigate whether the techniques used in previous hashing methods for single views can be generalised to the multiple view hashing problem. Fo example, the geometric information explored for the single view ANN is still useful as each view could be investigated separately and then the relationship between different views is combined for classification.

There are other interesting research questions for the single view hashing problem. For example, most of the hashing methods assume that the training data set has the same distribution as the base data sets. In real world data sets, this might not always the case due to the large volume of data or when the base data set is generated continually such as the images in social networks. Hashing methods, which adapt to this kind of change, might have better retrieval performance since they would incorporate the newest distribution information from the data base. To learn this kind of encoding function, we might consider two aspects of the optimisation model. The first one is that the training data should be sampled chronologically, i.e., the time information of the training data points should be considered for learning the encoding functions. The mini-batch stochastic gradient descent-like method used in this thesis might be useful for solving the corresponding optimisation problem since only a batch of data points are used in each iteration. The second issue is how to preserve the previously learned information. The encoding functions might loose the capability to preserve the geometric information of the whole base data set if we only use the newest training data set to update the encoding functions. Thus, both the learned information and the new training data set should be considered. This phenomenon is often referred as ‘catastrophic forgetting’ in training neural networks. For addressing this issue, the mechanisms of rehearsal (Robins, 1995) are the first priority for us to consider.

References

Absil, P. A., Mahony, R., and Sepulchre, R. (2009). *Optimization algorithms on matrix manifolds.* Princeton University Press.

Aggarwal, J. and Ryoo, M. (2011). Human activity analysis: A review. *ACM Computing Surveys*, *43*(3), 1–43.

Arias-Castro, E., Lerman, G., and Zhang, T. (2013). Spectral clustering based on local PCA. *arXiv preprint arXiv:1301.2007*.

Belkin, M. and Niyogi, P. (2003). Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, *15*(6), 1373–1396.

Bengio, Y. (2012). Practical recommendations for gradient-based training of deep architectures. *Neural Networks: Tricks of the Trade*, 437–478.

Bengio, Y., Paiement, J. f., Vincent, P., Delalleau, O., Roux, N. L., and Ouimet, M. (2004). Out-of-sample extensions for LLE, Isomap, MDS, Eigenmaps, and Spectral Clustering. In *Advances in Neural Information Processing Systems*, 177–184.

Bentley, J. L. (1975). Multidimensional binary search trees used for associative searching. *Communications of the ACM*, *18*(9), 509–517.

Berkhin, P. (2006). A survey of clustering data mining techniques. In *Grouping Multidimensional Data*, 25–71. Springer.

Brown, M. and Lowe, D. (2003). Recognising panoramas. In *Proceedings of International Conference on Computer Vision*, 1218–1225.

Cai, D., He, X., Zhang, W. V., and Han, J. (2007). Regularized locality preserving indexing via spectral regression. In *Proceedings of the 16th ACM Conference on Information and Knowledge Management*, 741–750.

Chaudhry, R., Ravichandran, A., Hager, G., and Vidal, R. (2009). Histograms of oriented optical flow and Binet-Cauchy kernels on nonlinear dynamical systems for the recognition of human actions. In *IEEE Conference on Computer Vision and Pattern Recognition*, 1932–1939.

Chen, G. and Lerman, G. (2009). Spectral curvature clustering (SCC). *International Journal of Computer Vision*, *81*(3), 317–330.

Cheng, J., Leng, C., Wu, J., Cui, H., and Lu, H. (2014). Fast and accurate image matching with cascade hashing for 3D reconstruction. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition.*

Chua, T. S., Tang, J., Hong, R., Li, H., Luo, Z., and Zheng, Y. (2009). NUS-WIDE: A real-world web image database from National University of Singapore. In *ACM International Conference on Image and Video Retrieval.*

Conway, J. H., Hardin, R. H., and Sloane, N. J. (1996). Packing lines, planes, etc.: Packings in Grassmannian spaces. *Experimental Mathematics*, *5*(2), 139–159.

Cox, T. and Cox, M. (1994). *Multidimensional scaling.* Chapman and Hall, London.

Dalal, N., Triggs, B., and Schmid, C. (2006). Human detection using oriented histograms of flow and appearance. In *Proceedings of European Conference on Computer Vision*, 428–441.

Danafar, S. and Gheissari, N. (2007). Action recognition for surveillance applications using optic flow and SVM. In *Proceedings of the 8th Asian Conference on Computer Vision*, 457–466.

Datar, M., Immorlica, N., Indyk, P., and Mirrokni, V. S. (2004). Locality-sensitive hashing scheme based on p-stable distributions. In *ACM Symposium on Computational Geometry*, 252–262.

De Lathauwer, L., De Moor, B., and Vandewalle, J. (2000). A multilinear singular value decomposition. *SIAM Journal on Matrix Analysis and Applications*, *21*(4), 1253–1278.

Donoho, D. L. (2000). Aide-memoire. High-dimensional data analysis: The curses and blessings of dimensionality. *American Math. Society Lecture-Math Challenges of the 21st Century.*

Donoho, D. L. and Grimes, C. (2003). Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data. *Proceedings of the National Academy of Sciences*, *100*(10), 5591–5596.

Drineas, P. and Mahoney, M. W. (2005). On the Nyström method for approximating a Gram matrix for improved kernel-based learning. *Journal of Machine Learning Research*, *6*, 2153–2175.

Efros, A. A., Berg, A. C., Mori, G., and Malik, J. (2003). Recognizing action at a distance. In *Proceedings of International Conference on Computer Vision*, 726–733.

Fletcher, P. T., Lu, C., Pizer, S. M., and Joshi, S. (2004). Principal geodesic analysis for the study of nonlinear statistics of shape. *IEEE Transactions on Medical Imaging*, *23*(8), 995–1005.

Frome, A., Singer, Y., Sha, F., and Malik, J. (2007). Learning globally-consistent local distance functions for shape-based image retrieval and classification. In *Proceedings of International Conference on Computer Vision*, 1–8.

Fu, X., Martin, S., Mills, S., and McCane, B. (2013). Improved spectral clustering using adaptive Mahalanobis distance. In *The 2nd Asian Conference on Pattern Recognition*, 171–175.

Fu, X., McCane, B., Albert, M., and Mills, S. (2013). Action recognition based on principal geodesic analysis. In *Proceedings of the 28th Conference on Image and Vision Computing New Zealand*, 259–264.

Fu, X., McCane, B., Mills, S., and Albert, M. (2014). NOKmeans: Non-orthogonal k-means hashing. In *Proceedings of the 12th Asian Conference on Computer Vision.*

Fu, X., McCane, B., Mills, S., and Albert, M. (2015). How to select hashing bits? A direct measurement approach. In *Proceedings of the 30th Conference on Image and Vision Computing New Zealand.*

Gallivan, K., Srivastava, A., Liu, X., and Dooren, P. (2003). Efficient algorithms for inferences on Grassmann manifolds. In *Proceedings of the 12th IEEE workshop on statistical signal processing*, 315–318.

Goldberg, A. B., Zhu, X., Singh, A., Xu, Z., and Nowak, R. (2009). Multi-manifold semi-supervised learning. In *International Conference on Artificial Intelligence and Statistics*, 169–176.

Gong, D., Zhao, X., and Medioni, G. (2012). Robust multiple manifolds structure learning. In *Proceedings of the 29th International Conference on Machine Learning*.

Gong, Y. and Lazebnik, S. (2011). Iterative quantization: A Procrustean approach to learning binary codes. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 817–824.

Griffin, G., Holub, A., and Perona, P. (2007). Caltech-256 object category dataset. *California Institute of Technology*.

Guttman, A. (1984). R-trees: A dynamic index structure for spatial searching. In *ACM SIGMOD Conference on Management of Data*.

Hamo, Y. and Markovitch, S. (2005). The COMPSET algorithm for subset selection. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, 728–733.

Harandi, M. T., Sanderson, C., Shirazi, S., and Lovell, B. C. (2011). Graph embedding discriminant analysis on Grassmannian manifolds for improved image set matching. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2705–2712.

He, K., Wen, F., and Sun, J. (2013). K-means hashing: An affinity-preserving quantization method for learning binary compact codes. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*.

He, X. and Niyogi, P. (2004). Locality preserving projections. In *Advances in Neural Information Processing Systems*, 153–160.

Hinton, G. E. and Roweis, S. T. (2002). Stochastic neighbor embedding. In *Advances in Neural Information Processing Systems*, 833–840.

Hinton, G. E. and Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science*, *313*(5786), 504–507.

Indyk, P. and Motwani, R. (1998). Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing*, 604–613.

Jain, P., Kulis, B., and Grauman, K. (2008). Fast image search for learned metrics. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*.

Jégou, H., Douze, M., and Schmid, C. (2008). Hamming embedding and weak geometric consistency for large scale image search. In *Proceedings of European Conference on Computer Vision*, 304–317.

Jégou, H., Douze, M., and Schmid, C. (2010). Improving bag-of-features for large scale image search. *International Journal of Computer Vision*, *87*(3), 316–336.

Jégou, H., Douze, M., and Schmid, C. (2011). Product quantization for nearest neighbor search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *33*(1), 117–128.

Jégou, H., Tavenard, R., Douze, M., and Amsaleg, L. (2011). Searching in one billion vectors: Re-rank with source coding. In *International Conference on Acoustics, Speech, and Signal Processing*, 861–864.

Jiang, Z., Lin, Z., and Davis, L. S. (2012). Recognizing human actions by learning and matching shape-motion prototype trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *34*(3), 533–547.

Jordan, M. I. and Mitchell, T. M. (2015). Machine learning: Trends, perspectives, and prospects. *Science*, *349*(6245), 255–260.

Karcher, H. (1977). Riemannian center of mass and mollifier smoothing. *Communications on Pure and Applied Mathematics*, *30*(5), 509–541.

Kim, K. I., Tompkin, J., and Theobalt, C. (2013). Curvature-aware regularization on Riemannian submanifolds. In *Proceedings of International Conference on Computer Vision*, 881–888.

Kim, T., Wong, S., and Cipolla, R. (2007). Tensor canonical correlation analysis for action classification. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 18–23.

Kim, T. K. and Cipolla, R. (2009). Canonical correlation analysis of video volume tensors for action categorization and detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *31*(8), 1415–1428.

Kirkpatrick, S., Gelatt, C. D., Vecchi, M. P., *et al.* (1983). Optimization by simulated annealing. *Science*, *220*(4598), 671–680.

Kulis, B. and Grauman, K. (2009). Kernelized locality-sensitive hashing for scalable image search. In *Proceedings of International Conference on Computer Vision.*

Kushnir, D., Galun, M., and Brandt, A. (2006). Fast multiscale clustering and manifold identification. *Pattern Recognition*, *39*(10), 1876–1891.

Laptev, I. and Lindeberg, T. (2003). Space-time interest points. In *Proceedings of International Conference on Computer Vision*, 432–439.

Lawrence, N. D. (2004). Gaussian process latent variable models for visualisation of high dimensional data. In *Advances in Neural Information Processing Systems*.

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

Lee, J. M. (2010). *Introduction to topological manifolds*, Volume 218. Springer-Verlag New York.

Lee, J. M. (2012). *Introduction to smooth manifolds*, Volume 202. Springer-Verlag New York.

Lin, B., He, X., Zhang, C., and Ji, M. (2013). Parallel vector field embedding. *Journal of Machine Learning Research*, *14*(1), 2945–2977.

Lin, Z., Jiang, Z., and Davis, L. S. (2009). Recognizing actions by shape-motion prototype trees. In *Proceedings of International Conference on Computer Vision*, 444–451.

Liu, D. C. and Nocedal, J. (1989). On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, *45*(1-3), 503–528.

Liu, W., Mu, C., Kumar, S., and Chang, S. F. (2014). Discrete graph hashing. In *Advances in Neural Information Processing Systems*.

Liu, W., Wang, J., Ji, R., Jiang, Y. G., and Chang, S. F. (2012). Supervised hashing with kernels. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2074–2081.

Liu, W., Wang, J., Kumar, S., and Chang, S. F. (2011). Hashing with graphs. In *Proceedings of International Conference on Machine Learning*.

Liu, X., He, J., Lang, B., and Chang, S. F. (2013). Hash bit selection: A unified solution for selection problems in hashing. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 1570–1577.

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, *60*(2), 91–110.

Lui, Y. (2012a). A least squares regression framework on manifolds and its application to gesture recognition. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, 13–18.

Lui, Y. (2012b). Tangent bundles on special manifolds for action recognition. *IEEE Transactions on Circuit and Systems for Video Technology*, *22*(6), 930–942.

Lui, Y. and Beveridge, J. R. (2011). Tangent bundle for human action recognition. In *IEEE International Conference on Automatic Face and Gesture Recognition*, 97–102.

Lui, Y., Beveridge, J. R., and Kirby, M. (2010). Action classification on product manifolds. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 833–839.

Lui, Y. M. (2012c). Advances in matrix manifolds for computer vision. *Image and Vision Computing*, *30*(6), 380–388.

Lui, Y. M. (2012d). Human gesture recognition on product manifolds. *Journal of Machine Learning Research*, *13*, 3297–3321.

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

Mu, Y., Hua, G., Fan, W., and Chang, S. F. (2014). Hash-SVM: Scalable kernel machines for large-scale visual classification. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition.*

Nene, S. A., Nayar, S. K., and Murase, H. (1996). Columbia object image library (COIL-20). Technical report, Technical Report CUCS-005-96.

Ng, A., Jordan, M., and Weiss, Y. (2002). On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems.*

Niebles, J. C., Wang, H., and Fei-Fei, L. (2008). Unsupervised learning of human action categories using spatial-temporal words. *International Journal of Computer Vision*, *79*(3), 299–318.

Norouzi, M. and Blei, D. M. (2011). Minimal loss hashing for compact binary codes. In *Proceedings of the 28th International Conference on Machine Learning*, 353–360.

Norouzi, M. and Fleet, D. (2013). Cartesian k-means. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 3017–3024.

Norouzi, M., Punjani, A., and Fleet, D. (2012). Fast search in hamming space with multi-index hashing. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 3108–3115.

Oliva, A. and Torralba, A. (2001). Modeling the shape of the scene: A holistic representation of the spatial envelope. *International Journal of Computer Vision*, *42*(3), 145–175.

Onbaşoğlu, E. and Özdamar, L. (2001). Parallel simulated annealing algorithms in global optimization. *Journal of Global Optimization*, *19*(1), 27–50.

O'Hara, S. (2013). *Scalable learning of actions from unlabeled videos.* Ph. D. thesis, Colorado State University.

Poppe, R. (2010). A survey on vision-based human action recognition. *Image and vision computing*, *28*(6), 976–990.

Rifai, S., Vincent, P., Muller, X., Glorot, X., and Bengio, Y. (2011). Contractive auto-encoders: Explicit invariance during feature extraction. In *Proceedings of the 28th International Conference on Machine Learning.*

Robins, A. (1995). Catastrophic forgetting, rehearsal and pseudorehearsal. *Connection Science: Journal of Neural Computing, Artificial Intelligence and Cognitive Research*, *7*(2), 123–146.

Rodriguez, A. and Laio, A. (2014). Clustering by fast search and find of density peaks. *Science*, *344*(6191), 1492–1496.

Roweis, S. T. and Saul, L. K. (2000). Nonlinear dimensionality reduction by locally linear embedding. *Science*, *290*(5500), 2323–2326.

Schuldt, C., Laptev, I., and Caputo, B. (2004). Recognizing human actions: A local SVM approach. In *Proceedings of 17th International Conference on Pattern Recognition*, 32–36.

Shechtman, E. and Irani, M. (2005). Space-time behavior based correlation. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 405–412.

Shi, J. and Malik, J. (2000). Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *22*(8), 888–905.

Shrivastava, A. and Li, P. (2014). Asymmetric LSH (ALSH) for sublinear time maximum inner product search (MIPS). In *Advances in Neural Information Processing Systems.*

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

Tipping, M. E. and Bishop, C. M. (1999). Mixtures of probabilistic principal component analyzers. *Neural Computation*, *11*(2), 443–482.

Torralba, A., Fergus, R., and Freeman, W. T. (2008). 80 million tiny images: A large data set for nonparametric object and scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *30*(11), 1958–1970.

Torralba, A., Fergus, R., and Weiss, Y. (2008). Small codes and large image databases for recognition. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 1–8.

Tron, R. and Vidal, R. (2007). A benchmark for the comparison of 3-d motion segmentation algorithms. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*.

Turaga, P., Veeraraghavan, A., and Chellappa, R. (2008). Statistical analysis on Stiefel and Grassmann manifolds with applications in computer vision. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 1–8.

Turaga, P., Veeraraghavan, A., Srivastava, A., and Chellappa, R. (2011). Statistical computations on Grassmann and Stiefel manifolds for image and video-based recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *33*(11), 2273–2286.

Van Der Maaten, L. and Hinton, G. (2008). Visualizing data using t-SNE. *Journal of Machine Learning Research*, *9*(85), 2579–2605.

Van Der Maaten, L. J., Postma, E. O., and Van Den Herik, H. J. (2009). Dimensionality reduction: A comparative review. *Journal of Machine Learning Research*, *10*(1-41), 66–71.

Vedaldi, A. and Fulkerson, B. (2008). VLFeat: An open and portable library of computer vision algorithms. In *Proceedings of the International Conference on Multimedia*, 1469–1472.

Vidal, R., Ma, Y., and Sastry, S. (2005). Generalized principal component analysis (GPCA). *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *27*(12), 1945–1959.

Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., and Manzagol, P. A. (2010). Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, *11*, 3371–3408.

Von Luxburg, U. (2007). A tutorial on spectral clustering. *Statistics and Computing*, *17*(4), 395–416.

Wang, J., Kumar, S., and Chang, S. (2010a). Semi-supervised hashing for scalable image retrieval. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 3424–3431.

Wang, J., Kumar, S., and Chang, S. F. (2010b). Sequential projection learning for hashing with compact codes. In *Proceedings of the 27th International Conference on Machine Learning*.

Wang, J., Wang, J., Yu, N., and Li, S. (2013). Order preserving hashing for approximate nearest neighbor search. In *Proceedings of the 21st ACM International Conference on Multimedia*, 133–142.

Wang, Y., Jiang, Y., Wu, Y., and Zhou, Z. H. (2011). Spectral clustering on multiple manifolds. *IEEE Transactions on Neural Networks*, *22*(7), 1149–1161.

Weber, R., Schek, H. J., and Blott, S. (1998). A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In *Proceedings of the 24th International Conference on Very Large Data Bases*, 194–205.

Weinberger, K. Q., Blitzer, J., and Saul, L. K. (2005). Distance metric learning for large margin nearest neighbor classification. In *Advances in Neural Information Processing Systems*, 1473–1480.

Weinberger, K. Q., Sha, F., and Saul, L. K. (2004). Learning a kernel matrix for nonlinear dimensionality reduction. In *Proceedings of the 21st International Conference on Machine Learning*, 106.

Weiss, Y., Fergus, R., and Torralba, A. (2012). Multidimensional spectral hashing. In *Proceedings of European Conference on Computer Vision*, 340–353.

Weiss, Y., Torralba, A., and Fergus, R. (2008). Spectral hashing. In *Advances in Neural Information Processing Systems*.

Xing, E. P., Jordan, M. I., Russell, S., and Ng, A. Y. (2002). Distance metric learning with application to clustering with side-information. In *Advances in Neural Information Processing Systems*, 505–512.

Xu, B., Bu, J., Lin, Y., Chen, C., He, X., and Cai, D. (2013). Harmonious hashing. In *International Joint Conference on Artificial Intelligence*.

Yan, S., Xu, D., Zhang, B., Zhang, H. J., Yang, Q., and Lin, S. (2007). Graph embedding and extensions: A general framework for dimensionality reduction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *29*(1), 40–51.

Yang, L. and Jin, R. (2006). Distance metric learning: A comprehensive survey. *Michigan State Universiy*.

Yu, X., Zhang, S., Liu, B., Zhong, L., and Metaxas, D. N. (2013). Large scale medical image search via unsupervised PCA hashing. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*.

Zhang, K., Zhang, L., and Yang, M. (2012). Real-time compressive tracking. In *Proceedings of European Conference on Computer Vision*, 864–877.

Zhang, Z. and Zha, H. (2004). Principal manifolds and nonlinear dimensionality reduction via tangent space alignment. *SIAM Journal on Scientific Computing*, *26*(1), 313–338.

Appendix A

Gradients calculation

A.1 Gradients calculation details

A.1.1 Objective function

The objective function for *Auto-JacoBin* is:

$$C(W_1, W_2, b_1, b_2) = \sum_{i=1}^{N}(||x_i - z_i||_F^2 + ||J_i - T_i T_i'||_F^2) + \alpha||YY' - NI||_1^{\epsilon}.$$

Here we focus on the gradients of C, for the objective functions of *AutoBin*, the only difference between the two is that the latter does not include the first order constraint. For efficient implementation in Matlab, all of the gradients are formulated in terms of matrix operations. In the following subsections, we present the gradients for each components according to W_1, b_1, W_2, b_2 respectively. For a batch of training data points, the final gradients is straightforward. Fig. A.1 displays the three layer neural network used in our method.

In the following gradients, $\odot$ is the operator of point-wise product between two matrices. To unveil the steps of the gradients calculation, we denote $W_2^{(i)}$ as the i^{th} row vector of W_2, 1_i^n as the zero column vector with the i^{th} position equal to 1, O^n as an n-dimensional one vector, and $\delta_{n,m}^{W_1}$ as the zero matrix with size W_1 with its $(n,m)^{\text{th}}$ position 1. We also use $\otimes$, $\ominus$ and $sum(\cdot)$ operators, which function as follows. The $W_1 \otimes W_2$ is a $D \times D \times d$ matrix such that its $(i,j,k)^{\text{th}}$ element is $W_1^{(ki)} W_2^{(jk)}$. For a $D \times D$ matrix U and a $D \times D \times d$ matrix V, $U \ominus V$ is a $D \times D \times d$ matrix such that its $(i,j,k)^{\text{th}}$ element is $U^{(ij)} V^{(ijk)}$. The operation of both $\otimes$ and $\ominus$ can be implemented in Matlab by the $repmat$ and dot functions efficiently. For a 2-D or 3-D matrix Q, the $sum(Q)$ produces respectively a 1-D or 2-D matrix by adding the row entries similarly to the sum function in Matlab.

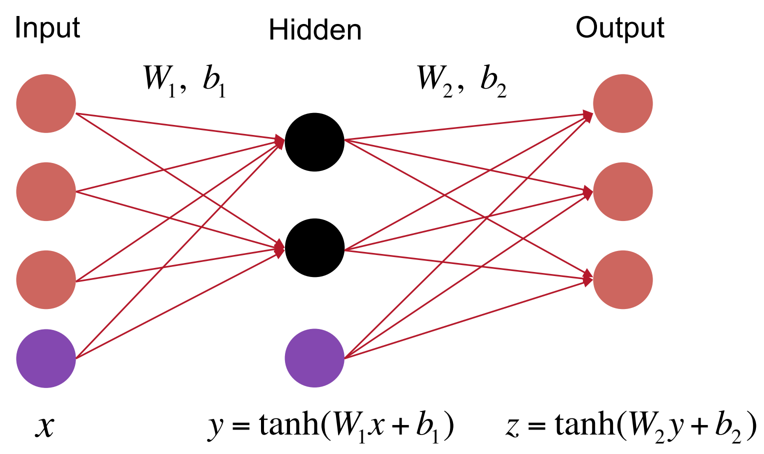


Fig. A.1. The Auto-encoder model used in our proposed method. The hidden layer is colored in black, the biases in purple. The forward calculation is shown under the corresponding layer.

A.1.2 Auto-encoder constraint

The gradients can be calculated as:

$$\frac{\partial ||z-x||_F^2}{\partial W_1} = 2\left(\left(W_2'\left((z-x)\odot\left(1-z^2\right)\right)\right)\odot\left(1-y^2\right)\right)x'$$

$$\frac{\partial ||z-x||_F^2}{\partial b_1} = 2\left(\left(W_2'\left((z-x)\odot\left(1-z^2\right)\right)\right)\odot\left(1-y^2\right)\right)$$

$$\frac{\partial ||z-x||_F^2}{\partial W_2} = 2\left((z-x)\odot\left(1-z^2\right)\right)y'$$

$$\frac{\partial ||z-x||_F^2}{\partial b_2} = 2\left((z-x)\odot\left(1-z^2\right)\right)$$

A.1.3 Binary constraint

$$\frac{\partial ||YY'-NI||_1^\epsilon}{\partial W_1} = 2\left(\left((YY'-NI)\left((YY'-NI)^2+\epsilon\right)^{-\frac{1}{2}}Y\right)\odot\left(1-Y^2\right)\right)X'$$
$$\frac{\partial ||YY'-NI||_1^\epsilon}{\partial b_1} = 2\left(\left((YY'-NI)\left((YY'-NI)^2+\epsilon\right)^{-\frac{1}{2}}Y\right)\odot\left(1-Y^2\right)\right)$$

A.1.4 First order constraint

To simplify the notation, we denote $A = TT'$. Since the constraint can be decomposed as $||J - A||_F^2 = ||J||_F^2 - 2\,\mathrm{Tr}(JA') + const$, the gradients can be decomposed into two components:

Gradients for $||J||_F^2$

Since z is a vector, we can calculate its gradient according to each component $z^{(i)}$:

$$\frac{\partial(z^{(i)})}{\partial x} = \frac{\partial(\tanh(W_2^{(i)}\tanh(W_1 x + b_1) + b_2^{(i)}))}{\partial x} = W_1'\left(\left(\left(W_2^{(i)}\right)'(1-(z^{(i)})^2)\right)\odot\left(1-y^2\right)\right)$$

The Jacobian function can be expressed as

$$J = W_1'\left(W_2' \odot \left(1-y^2\right)\left(1-z^2\right)'\right)$$

and

$$||J||_F^2 = ||\frac{\partial(z)}{\partial x}||_F^2 = \sum_{i=1}^{D}\left(1-(z^{(i)})^2\right)^2\sum_{m=1}^{D}\left(\sum_{n=1}^{d}W_1^{(n,m)}W_2^{(i,n)}\left(1-(y^{(n)})^2\right)\right)^2$$

Thus

$$\begin{aligned}
\frac{\partial||J||_F^2}{\partial W_1} &= \sum_{i=1}^{D}\frac{\partial\left(1-(z^{(i)})^2\right)^2}{\partial W_1}\sum_{m=1}^{D}\left(\sum_{n=1}^{d}W_1^{(n,m)}W_2^{(i,n)}\left(1-(y^{(n)})^2\right)\right)^2 \\
&\quad+\sum_{i=1}^{D}\left(1-(z^{(i)})^2\right)^2\sum_{m=1}^{D}\frac{\partial\left(\sum_{n=1}^{d}W_1^{(n,m)}W_2^{(i,n)}\left(1-(y^{(n)})^2\right)\right)^2}{\partial W_1} \\
&=\sum_{i=1}^{D}\left(\left(\left(W_2^{(i)}\right)'2\left(1-(z^{(i)})^2\right)^2\left(-2z^{(i)}\right)\right)\odot\left(1-y^2\right)\right)x'\sum_{m=1}^{D}\left(\sum_{n=1}^{d}W_1^{(n,m)}W_2^{(i,n)}\left(1-(y^{(n)})^2\right)\right)^2 \\
&\quad+\sum_{i=1}^{D}\left(1-(z^{(i)})^2\right)^2\sum_{m=1}^{D}2\left(\sum_{n=1}^{d}W_1^{(n,m)}W_2^{(i,n)}\left(1-(y^{(n)})^2\right)\right)\left(\sum_{n=1}^{d}\delta_{n,m}^{W_1}W_2^{(i,n)}\left(1-(y^{(n)})^2\right)\right) \\
&\quad+\sum_{i=1}^{D}\left(1-(z^{(i)})^2\right)^2\sum_{m=1}^{D}2\left(\sum_{n=1}^{d}W_1^{(n,m)}W_2^{(i,n)}\left(1-(y^{(n)})^2\right)\right) \\
&\quad\times\left(\sum_{n=1}^{d}1_n^d\left(W_1^{(n,m)}W_2^{(i,n)}\left(-2y^{(n)}\right)\left(1-(y^{(n)})^2\right)\right)x'\right) \\
&=-4\left(W_2'\odot\left(O^d\left(\left(1-z^2\right)^2(z)\right)'\right)\odot\left(\left(1-y^2\right)\left(O^D\right)'\right)\odot\left(O^d sum\left(\left(W_1'\left(W_2'\odot\left(1-y^2\right)(O^D)'\right)\right)^2\right)\right)\right)O^D x' \\
&\quad+2\left(W_2'\left(W_1'\left(W_2'\odot\left(O^d\left(\left(1-z^2\right)^2\right)'\right)\odot\left(\left(1-y^2\right)(O^D)'\right)\right)\right)'\right)\odot\left(\left(1-y^2\right)\left(O^D\right)'\right) \\
&\quad-4\left(sum\left(sum\left(\left(\left(O^D\left(\left(1-z^2\right)^2\right)'\right)\odot\left(W_1'\left(W_2'\odot\left(1-y^2\right)\left(O^D\right)'\right)\right)\right)\ominus\left(W_1\otimes W_2\right)\right)\right)\right)' \\
&\quad\odot\left(y-y^3\right)x'
\end{aligned}$$

$$\begin{aligned}
\frac{\partial ||J||_F^2}{\partial b_1} &= \sum_{i=1}^{D} \frac{\partial \left(1-(z^{(i)})^2\right)^2}{\partial b_1} \sum_{m=1}^{D} \left(\sum_{n=1}^{d} W_1^{(n,m)} W_2^{(i,n)} \left(1-(y^{(n)})^2\right) \right)^2 \\
&\quad + \sum_{i=1}^{D} \left(1-(z^{(i)})^2\right)^2 \sum_{m=1}^{D} \frac{\partial \left(\sum_{n=1}^{d} W_1^{(n,m)} W_2^{(i,n)} \left(1-(y^{(n)})^2\right) \right)^2}{\partial b_1} \\
&= \sum_{i=1}^{D} \left(\left(\left(W_2^{(i)}\right)' 2 \left(1-(z^{(i)})^2\right)^2 \left(-2z^{(i)}\right) \right) \odot \left(1-y^2\right) \right) \sum_{m=1}^{D} \left(\sum_{n=1}^{d} W_1^{(n,m)} W_2^{(i,n)} \left(1-(y^{(n)})^2\right) \right)^2 \\
&\quad + \sum_{i=1}^{D} \left(1-(z^{(i)})^2\right)^2 \sum_{m=1}^{D} 2 \left(\sum_{n=1}^{d} W_1^{(n,m)} W_2^{(i,n)} \left(1-(y^{(n)})^2\right) \right) \\
&\quad \times \left(\sum_{n=1}^{d} \left(W_1^{(n,m)} W_2^{(i,n)} \left(-2y^{(n)}\right) \left(1-(y^{(n)})^2\right) \right) 1_n^D \right) \\
&= -4 \left(W_2' \odot \left(O^d \left(\left(1-z^2\right)^2 (z) \right)' \right) \odot \left(\left(1-y^2\right) \left(O^D\right)' \right) \odot \left(O^d sum \left(\left(W_1' \left(W_2' \odot \left(1-y^2\right) (O^D)' \right) \right)^2 \right) \right) \right) O^D \\
&\quad - 4 \left(sum \left(sum \left(\left(\left(O^D \left(\left(1-z^2\right)^2 \right)' \right) \odot \left(W_1' \left(W_2' \odot \left(1-y^2\right) \left(O^D\right)' \right) \right) \right) \ominus \left(W_1 \otimes W_2\right) \right) \right) \right)' \\
&\quad \odot \left(y-y^3\right)
\end{aligned}$$

$$\begin{aligned}
\frac{\partial ||J||_F^2}{\partial W_2} &= \sum_{i=1}^{D} \frac{\partial \left(1-(z^{(i)})^2\right)^2}{\partial W_2} \sum_{m=1}^{D} \left(\sum_{n=1}^{d} W_1^{(n,m)} W_2^{(i,n)} \left(1-(y^{(n)})^2\right) \right)^2 \\
&\quad + \sum_{i=1}^{D} \left(1-(z^{(i)})^2\right)^2 \sum_{m=1}^{D} \frac{\partial \left(\sum_{n=1}^{d} W_1^{(n,m)} W_2^{(i,n)} \left(1-(y^{(n)})^2\right) \right)^2}{\partial W_2} \\
&= \sum_{i=1}^{D} 1_i^D 2(1-(z^{(i)})^2)^2(-2z^{(i)})y' \sum_{m=1}^{D} \left(\sum_{n=1}^{d} W_1^{(n,m)} W_2^{(i,n)} \left(1-(y^{(n)})^2\right) \right)^2 \\
&\quad + \sum_{i=1}^{D} \left(1-(z^{(i)})^2\right)^2 \sum_{m=1}^{D} 2 \left(\sum_{n=1}^{d} W_1^{(n,m)} W_2^{(i,n)} \left(1-(y^{(n)})^2\right) \right) \left(1_i^D \left(W_1^{(mc)} \odot \left(1-y^2\right) \right)' \right) \\
&= -4 \left(\left((1-z^2)^2 z\right) \odot \left(sum \left(\left(W_1' \left(W_2' \odot \left(1-y^2\right) (O^D)' \right) \right)^2 \right) \right)' \right) y' \\
&\quad + 2 \left(\left(1-z^2\right)^2 (O^d)' \right) \odot \left(\left(W_1' \left(W_2' \odot \left(1-y^2\right) (O^D)' \right) \right)' \left(W_1 \odot \left(1-y^2\right) (O^D)' \right)' \right)
\end{aligned}$$

$$\begin{aligned}
\frac{\partial ||J||_F^2}{\partial b_2} &= \sum_{i=1}^{D} \frac{\partial \left(1-(z^{(i)})^2\right)^2}{\partial b_2} \sum_{m=1}^{D} \left(\sum_{n=1}^{d} W_1^{(n,m)} W_2^{(i,n)} \left(1-(y^{(n)})^2\right) \right)^2 \\
&= \sum_{i=1}^{D} 2(1-(z^{(i)})^2)^2(-2z^{(i)}) 1_i^D \sum_{m=1}^{D} \left(\sum_{n=1}^{d} W_1^{(n,m)} W_2^{(i,n)} \left(1-(y^{(n)})^2\right) \right)^2 \\
&= -4 \left((1-z^2)^2 z\right) \odot \left(sum \left(\left(W_1' \left(W_2' \odot \left(1-y^2\right) (O^D)' \right) \right)^2 \right) \right)'
\end{aligned}$$

Gradients for $\mathrm{Tr}(JA')$

$$\mathrm{Tr}(JA') = \mathrm{Tr}(\frac{\partial(z)}{\partial x} A') = \sum_{i=1}^{D} \left(1-(z^{(i)})^2\right) \sum_{m=1}^{D} a_{mi} \left(\sum_{n=1}^{d} W_1^{(n,m)} W_2^{(i,n)} \left(1-(y^{(n)})^2\right) \right)$$

$$
\begin{aligned}
\frac{\partial \operatorname{Tr}(JA')}{\partial W_1} &= \sum_{i=1}^{D} \frac{\partial \left(1-(z^{(i)})^2\right)}{\partial W_1} \sum_{m=1}^{D} a_{mi} \left(\sum_{n=1}^{d} W_1^{(n,m)} W_2^{(i,n)} \left(1-(y^{(n)})^2\right) \right) \\
&\quad + \sum_{i=1}^{D} \left(1-(z^{(i)})^2\right) \frac{\partial \sum_{m=1}^{D} a_{mi} \left(\sum_{n=1}^{d} W_1^{(n,m)} W_2^{(i,n)} \left(1-(y^{(n)})^2\right) \right)}{\partial W_1} \\
&= \sum_{i=1}^{D} \left(\left(\left(W_2^{(i)}\right)' \left(1-(z^{(i)})^2\right) \left(-2z^{(i)}\right) \right) \odot \left(1-y^2\right) \right) x' \sum_{m=1}^{D} a_{mi} \left(\sum_{n=1}^{d} W_1^{(n,m)} W_2^{(i,n)} \left(1-(y^{(n)})^2\right) \right) \\
&\quad + \sum_{m=1}^{D} a_{mi} \left(\sum_{n=1}^{d} \delta_{n,m}^{W_1} W_2^{(i,n)} \left(1-(y^{(n)})^2\right) + 1_n^d \left(W_1^{(n,m)} W_2^{(i,n)} \left(-2y^{(n)}\right) \left(1-(y^{(n)})^2\right) \right) x' \right) \\
&\quad \sum_{i=1}^{D} \left(1-(z^{(i)})^2\right) \\
&= -2 \left(W_2' \odot O^d \left((z-z^3)\right)' \odot \left(1-y^2\right) O^D \odot O^d sum \left(\left(W_1' \left(W_2' \odot \left(1-y^2\right) \left(O^D\right)' \right) \right) \odot A \right) \right) O^D x' \\
&\quad + \left(\left(O^d \left(1-z^2\right)' \right) \odot \left(W_2' \odot \left(\left(1-y^2\right) \left(O^D\right)' \right) \right) \right) A' \\
&\quad - 2 \left(sum \left(sum \left(\left(\left(O^D \left(1-z^2\right) \right) \odot A \right) \ominus \left(W_1 \otimes W_2\right) \right) \right) \right)' \odot \left(y-y^3\right) x'
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial \operatorname{Tr}(JA')}{\partial b_1} &= \sum_{i=1}^{D} \frac{\partial \left(1-(z^{(i)})^2\right)}{\partial b_1} \sum_{m=1}^{D} a_{mi} \left(\sum_{n=1}^{d} W_1^{(n,m)} W_2^{(i,n)} \left(1-(y^{(n)})^2\right) \right) \\
&\quad + \sum_{i=1}^{D} \left(1-(z^{(i)})^2\right) \frac{\partial \sum_{m=1}^{D} a_{mi} \left(\sum_{n=1}^{d} W_1^{(n,m)} W_2^{(i,n)} \left(1-(y^{(n)})^2\right) \right)}{\partial b_1} \\
&= \sum_{i=1}^{D} \left(\left(\left(W_2^{(i)}\right)' \left(1-(z^{(i)})^2\right) \left(-2z^{(i)}\right) \right) \odot \left(1-y^2\right) \right) \sum_{m=1}^{D} a_{mi} \left(\sum_{n=1}^{d} W_1^{(n,m)} W_2^{(i,n)} \left(1-(y^{(n)})^2\right) \right) \\
&\quad + \sum_{i=1}^{D} \left(1-(z^{(i)})^2\right) \sum_{m=1}^{D} a_{mi} \left(1_n^d \left(W_1^{(n,m)} W_2^{(i,n)} \left(-2y^{(n)}\right) \left(1-(y^{(n)})^2\right) \right) \right) \\
&= -2 \left(W_2' \odot O^d \left((z-z^3)\right)' \odot \left(1-y^2\right) O^D \odot O^d sum \left(\left(W_1' \left(W_2' \odot \left(1-y^2\right) \left(O^D\right)' \right) \right) \odot A \right) \right) O^D \\
&\quad - 2 \left(sum \left(sum \left(\left(\left(O^D \left(1-z^2\right) \right) \odot A \right) \ominus \left(W_1 \otimes W_2\right) \right) \right) \right)' \odot \left(y-y^3\right)
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial \operatorname{Tr}(JA')}{\partial W_2} &= \sum_{i=1}^{D} \frac{\partial \left(1-(z^{(i)})^2\right)}{\partial W_2} \sum_{m=1}^{D} a_{mi} \left(\sum_{n=1}^{d} W_1^{(n,m)} W_2^{(i,n)} \left(1-(y^{(n)})^2\right) \right) \\
&\quad + \sum_{i=1}^{D} \left(1-(z^{(i)})^2\right) \frac{\partial \sum_{m=1}^{D} a_{mi} \left(\sum_{n=1}^{d} W_1^{(n,m)} W_2^{(i,n)} \left(1-(y^{(n)})^2\right) \right)}{\partial W_2} \\
&= \sum_{i=1}^{D} 1_i^D (1-(z^{(i)})^2)(-2z^{(i)}) y' \sum_{m=1}^{D} a_{mi} \left(\sum_{n=1}^{d} W_1^{(n,m)} W_2^{(i,n)} \left(1-(y^{(n)})^2\right) \right) \\
&\quad + \sum_{i=1}^{D} \left(1-(z^{(i)})^2\right) \sum_{m=1}^{D} a_{mi} \left(1_i^D \left(W_1^{(mc)} \odot \left(1-y^2\right) \right)' \right) \\
&= -2 \left(\left(z-z^3\right) \odot \left(sum \left(\left(W_1' \left(W_2' \odot \left(1-y^2\right) \left(O^D\right)' \right) \right) \odot A \right) \right)' \right) y' \\
&\quad + \left(\left(1-z^2\right) \left(O^d\right)' \right) \odot \left(\left(\left(W_1 \odot \left(1-y^2\right) \left(O^D\right)' \right) A \right)' \right)
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial \operatorname{Tr}(JA')}{\partial b_2} &= \sum_{i=1}^{D} \frac{\partial \left(1-(z^{(i)})^2\right)}{\partial b_2} \sum_{m=1}^{D} a_{mi} \left(\sum_{n=1}^{d} W_1^{(n,m)} W_2^{(i,n)} \left(1-(y^{(n)})^2\right) \right) \\
&= \sum_{i=1}^{D} 1_i^D (1-(z^{(i)})^2)(-2z^{(i)}) \sum_{m=1}^{D} a_{mi} \left(\sum_{n=1}^{d} W_1^{(n,m)} W_2^{(i,n)} \left(1-(y^{(n)})^2\right) \right) \\
&= -2 \left(\left(z-z^3\right) \odot \left(sum \left(\left(W_1' \left(W_2' \odot \left(1-y^2\right) \left(O^D\right)' \right) \right) \odot A \right) \right) \right)'
\end{aligned}
$$