Nguyen, Hieu (2011) Linear subspace methods in face recognition. PhD thesis, University of Nottingham.

Linear Subspace Methods in Face Recognition

Hieu V. Nguyen, BSc, MSc.

Thesis submitted to The University of Nottingham

for the degree of Doctor of Philosophy

November 2011

To my family...

Abstract

Despite over 30 years of research, face recognition is still one of the most difficult problems in the field of Computer Vision. The challenge comes from many factors affecting the performance of a face recognition system: noisy input, training data collection, speed-accuracy trade-off, variations in expression, illumination, pose, or ageing. Although relatively successful attempts have been made for special cases, such as frontal faces, no satisfactory methods exist that work under completely unconstrained conditions. This thesis proposes solutions to three important problems: lack of training data, speed-accuracy requirement, and unconstrained environments.

The problem of lacking training data has been solved in the worst case: single sample per person. Whitened Principal Component Analysis is proposed as a simple but effective solution. Whitened PCA works well under this scenario because of two reasons. On one hand, PCA has the potential to extract discriminating information as covariance matrix has characterised all the inherent differences in the training data. On the other hand, whitening process can exclude the trained variation retained by the PCA which is harmful to the recognition process. Whitened PCA performs consistently well on multiple face datasets.

Speed-accuracy trade-off problem is the second focus of this thesis. Accuracy is not the only requirement in face recognition systems. In many real world situations, such as video surveillance, speed plays an important role. Two solutions are proposed to tackle this problem. The first solution is a new feature extraction method called Compact Binary Patterns. Compact Binary Patterns is a more compact and accurate generalisation of Local Binary Patterns. Because of its compactness, Compact Binary Patterns is about three times faster than Local Binary Patterns. The second solution is a multi-patch classifier which combines multiple classifiers linearly using Memetic Algorithm. The resulting ensemble performs much better than a single classifier without compromising speed.

Two metric learning methods are introduced to solve the problem of unconstrained face recognition. The first method called Indirect Neighourhood Component Analysis combines the best ideas from Neighourhood Component Analysis and One-shot learning. The second method, Cosine Similarity Metric Learning, uses Cosine Similarity instead of the more popular Euclidean distance to form the objective function in the learning process. Because of the use of Cosine Similarity, the objective function is simpler thus faster to optimise. This Cosine Similarity Metric Learning method produces the best result in the literature on the state-of-the-art face dataset: the Labelled Faces in the Wild dataset.

Finally, a full face verification system based on our real experience taking part in ICPR 2010 Face Verification contest is described. Many practical points are discussed.

List of Publications

Some parts of the work presented in the thesis have been published in the following articles:

1. Hieu V. Nguyen, Li Bai. Cosine Similarity Metric Learning for Face Verification. *Asian Conference on Computer Vision*. ACCV 2010.

2. Hieu V. Nguyen, Li Bai. Face Verification Using Indirect Neighbourhood Components Analysis. *International Symposium on Visual Computing*. ISVC 2010.

3. Hieu V. Nguyen, Li Bai. Compact Binary Patterns (CBP) with Multiple Patch Classifiers for Fast and Accurate Face Recognition. *Computational Modeling of Objects Presented in Images: Fundamentals, Methods, and Applications*. CompIMAGE 2010 (best student paper award).

4. Hieu V. Nguyen, Li Bai, LinLin Shen. Local Gabor Binary Pattern Whitened PCA: A Novel Approach for Face Recognition from Single Image Per Person. *IAPR/IEEE International Conference on Biometrics*. ICB 2009.

Acknowledgements

I would like to thank many great people who have helped me over the course of my PhD degree. Without their help and support, it would have been much more difficult, if not impossible, for me to complete the course and this thesis.

First of all, I would like to thank my supervisor, Dr. Li Bai, for her guidance, patience and advice through the whole course of my research. I specially thank Dr. Bai for sending me to Shenzhen to collaborate with Dr. LinLin Shen of Shenzhen University in 2008. Through that experience, I learnt a tremendous amount of new things about the field of face recognition.

I would like to thank Dr. LinLin Shen for his time and support when I was visiting Shenzhen University. His expert knowledge about face recognition helped me to be much more productive. This experience alone speeded up my initial research progress at least three times.

I also thank Dr. Guoping Qiu, who gave me many interesting suggestions when I ran out of ideas.

I would also thank Dr. Yan Wang for teaming up with me to participate in the ICPR 2010 Face Verification contest. His eye detection algorithm worked really well and

was one of the key components in our system. I would like to thank my colleagues and friends in Intelligent Modelling and Analysis research group at the University of Nottingham. In particular I would like to thank Dr. Jan Feyereisl for reviewing my ACCV paper, which took a great amount of his time.

I would like to thank my father, my mother, and my sister for their support and encouragement through out many years since I was a child.

Finally, this thesis is for my wife, Ha Hoang, and my daughter, Lam Nguyen, who are always there to support me unconditionally.

Contents

List of Figures

List of Tables

Definition of Vocabulary

The following are informal definitions of important terminologies in the field of face recognition.

- Face Recognition: Face Identification or Face Verification.
- Face Identification: the task of identifying the identity of a given subject in a database.
- Face Verification: the task of deciding whether two faces belong to the same person or not.
- Training Data: a set of data samples which allows computers to learn from.
- Unsupervised Learning: a class of problems in which one seeks to determine how the data are organised.
- Supervised Learning: the task of inferring a function from supervised training data. In supervised learning, each example is a pair consisting of an input object (typically a vector) and a desired output value (also called the supervisory signal).

- Labelled and unlabelled training sample: if the class of a training sample is provided, that sample is called labelled, otherwise it is unlabelled.
- Labelled (unlabelled) training data: the set of labelled (unlabelled) training samples.

CHAPTER 1

Introduction

1.1 Objective

The ultimate objective in face recognition is to develop a system which works reliably under unconstrained conditions, runs quickly and requires minimum training data. Unfortunately, even after more than three decades, that objective is still far from being achieved. In this thesis, the grand objective is decomposed into three smaller objectives which can be achieved separately.

Our first objective is to deal with the problem of lacking training data. In real world situations, it is often costly to collect a large amount of training samples. In some cases such as passport or identification cards, it is even impossible to have more than one sample per person. The problem of lacking training samples can lead to overfitting problem in many learning algorithms. This can decrease the accuracy or even prevent these algorithms from success. It is desirable to achieve good performance in the worst case: single sample per person.

The second objective is to improve the accuracy and speed of current methods. Accuracy is not the only requirement in face recognition systems. In some real application like video surveillance, speed plays an important role. It is unacceptable if it takes 30 seconds to process a face in a video surveillance system. Ideally, the processing time should be less than the time between two consecutive input faces. In practice, there is often a trade-off between accuracy and speed. Therefore, the second objective of this thesis is to develop methods that are both accurate and fast.

The third, also the most ambitious, objective is to be able to recognise human faces within images reliably under completely unconstrained environments. Despite the effort of researchers over a couple of decades, this problem has remained unsolved for the most part. Although relatively successful attempts have been made for special cases, such as frontal faces, no satisfactory methods exist that work under unconstrained conditions. This thesis develops methods that can automatically recognise faces without any prior knowledge about face image data.

Since it is highly challenging to achieve all three objectives at the same time, each objective will be pursued separately.

1.2 Motivation

Despite being a challenging task for the computer, face recognition seems much easier for human beings. The ability to recognise faces is one of the most important human abilities. Face perception is a routine task for humans and is an important part of the capability of human perception system. It is widely believed that one can instantly recognise thousands of people with whom one is familiar. As with many perceptual

abilities, the ease with which humans can recognise faces disguises the complexity of the task. Hopefully, building successful face recognition systems can help us understand more about how human perception system works.

Also, being one of the most important applications in computer vision, face recognition has received significant attention in the last three decades. There are two main reasons to explain this popularity. The first is the wide range of commercial and law enforcement applications. Some examples of face recognition applications are:

- Entertainment: video games, virtual reality, training programs, human robot interaction, human computer interaction.
- Smart cards: drivers' licenses, entitlement programs, immigration, national ID, passports, voter registration, welfare fraud.
- Information security: TV parental control, personal device logon, desktop logon, application security, database security, file encryption, intranet security, internet access, medical records, secure trading terminals.
- Law enforcement: advanced video surveillance, CCTV control and surveillance, portal control, post-event analysis, shoplifting, suspect tracking and investigation.

The second reason is the advantages of using faces over other biometric indicators. Face recognition is not the only option in the aforesaid applications. There are even more reliable methods for biometric personal recognition but face recognition offers unique advantages. For example, fingerprint analysis and iris scans rely on the cooperation of the participants whereas a personal identification system based on analysis of frontal

or profile images of the face is often effective without the participant's cooperation or knowledge. In other words, face recognition is non-intrusive to users and this is extremely important to deploy applications widely in practice. As a result, the problem of automatic recognition of human faces continues to attract researchers from many disciplines such as image processing, pattern recognition, neural networks, computer vision, computer graphics, and psychology.

While there are many compelling reasons to use face recognition, no practical solutions exist. This is due to the difficulties inherent in the problem, which is now examined.

1.3 Challenge

Successful approaches to face recognition need to address a variety of problems: high dimensional space, lack of training data, noisy input, variations of expression, illumination, pose, and even ageing.

For the purpose of classification, images are often transformed to vectors in a high dimensional space. A simple method is to consider the intensity of each pixel to be a single component of the vector. The number of dimensions is the same as the number of pixels in an image, i.e., if each image has the sizes of 100×100 then the dimension is 10,000, which is often a very big number. This representation simplifies the application of computer vision algorithms, such as k-nearest neighbour, but it also increases the processing time considerably. In addition, it can cause the curse of dimensionality problem that makes the computation intractable. This very first challenge directly relates to the problem of lacking training data.

Modern learning methods require a large amount of data in order to produce good

Figure 1.1: Facial Variations: noise (b), expression (c), illumination (d), pose (e), and ageing (f) (*taken from* [1])

results. For a 100×100 face image being vectorised into a 10,000 dimensions feature space, theoretically the number of training images for each person should be at least ten times that of the dimensionality [6], that is, 100,000 images in total per person. Intuitively, it is hard to imagine that human beings need so many photographs of a person in order to develop a good model of his appearance. The number of training samples required will be a lot greater if noise and all possible variations in face images: expression, illumination, pose, and ageing is considered.

It has been observed that the variations between the images of the same face due to expression, illumination and viewing direction are almost always larger than variations from the change in face identity [7]. As shown in Figure 1.1, the same person can appear noticeably different when light source direction and viewpoint vary. These variations are increased by additional factors such as facial expression, hair styles, cos-

metics, and even changes due to ageing. This appearance variability makes it difficult to extract the inherent information, i.e. identity, of the face objects from their respective images. Under unconstrained environments, these variations are greatest, thus hardest to model.

1.4 Contribution

The major contributions of this thesis can be summarised as follows:

- Whitened Principal Component Analysis (Whitened PCA) is proposed to solve the problem of lacking training data. Our extensive experimental results show that Whitened PCA is robust to the lack of training data even in the worst scenario: only single sample per person.

- Two solutions are proposed to tackle the problems of high dimensional space and speed-accuracy trade-off. The first is a new feature extraction method called Compact Binary Patterns. Compact Binary Patterns is a generalised and improved version of Local Binary Patterns. Especially, Compact Binary Patterns has more discriminant power and runs three times faster than Local Binary Patterns. The second solution is a multi-patch classifier which is a linear ensemble of multiple classifiers created from different regions on the face. The ensemble performs much better than a single classifier without compromising the speed. Multiple experiments on FERET and ORL datasets show that our solutions improve both accuracy and speed considerably.

- A metric learning method called Indirect Neighbourhood Component Analysis

(INCA) is proposed to resolve the problem of unconstrained face recognition. This is the result of combining ideas from two recently introduced methods: One-shot learning (OSL) [8] and Neighbourhood Components Analysis (NCA) [9]. NCA is specifically designed for classification so it is not directly applicable to verification. OSL is designed for verification but does not fully utilise training data. Our method fixes these shortcomings to solve the verification problem and fully utilise training data at the same time. Experiments on the LFW dataset show that INCA performs well in real world situations.

- Another metric learning method called Cosine Similarity Metric Learning (CSML) is proposed to resolve the problem of unconstrained face recognition. This is a general metric learning method utilising Cosine Similarity as distance measure. Cosine Similarity will be shown to fit the metric learning context better than the more popular Euclidean distance, leading to a simpler objective function. Applying CSML to face verification, we produce the best results on the state-of-the-art dataset, the Labelled Faces in the Wild [10] in the literature.

- Last but not least, a complete face verification system is presented. It illustrates many ideas in this thesis in a practical context which was our real experience taking part in ICPR 2010 Face Verification contest. Our result was among the top submissions.

1.5 Outline of the thesis

The structure of this thesis is as follows.

In chapter 2, existing work in the field of face recognition focusing on feature extrac-

tion, classifier combination and subspace learning techniques is reviewed. These three topics are most relevant to the contribution of the thesis.

Chapter 3 presents our approach, Whitened Principal Component Analysis, to solve the problem of lacking training data.

In chapter 4, a new type of feature extraction method called Compact Binary Patterns is introduced. Chapter 4 also introduces another approach to increase the accuracy: using multiple patches to form an ensemble of classifiers. Memetic Algorithm is proposed to use to estimate weights of individual classifier because it is very fast to train and has strong optimisation power.

Chapter 5 introduces a novel metric learning method called Indirect Neighbourhood Components Analysis.

In chapter 6, another novel metric learning method called Cosine Similarity Metric Learning is introduced to solve completely unconstrained face recognition problem. This method achieved the best results in the literature on the state-of-the-art dataset Labelled Faces in the Wild [1].

In chapter 7, by putting everything together, we illustrate a complete face verification system which is also our real experience participating in ICPR 2010 Face Verification contest.

Finally, in chapter 8, conclusions and discuss future works are drawn.

CHAPTER 2

Literature review

2.1 General framework

Figure 2.1 presents a general framework for a face recognition system which consists of a number of steps ranging from face detection to classifier fusion.

- Face Detection: before a face can be recognised, it needs to be detected first. Face detection is the task of locating the position and size of a face in arbitrary images.
- Normalisation (or preprocessing): after the face is detected, it is often useful to normalise it. The purpose of this step is to make the condition of the processing face as close as possible with ones stored in the database. Two faces look different for many reasons other than their identities such as illumination, scale, or angle between two eyes. Normalisation can reduce these types of differences. As a result, the more important difference between two faces is intensified.
- Feature Extraction: this step is to extract useful features from the face. The output of this step is a vector storing information about the identity of the face. There

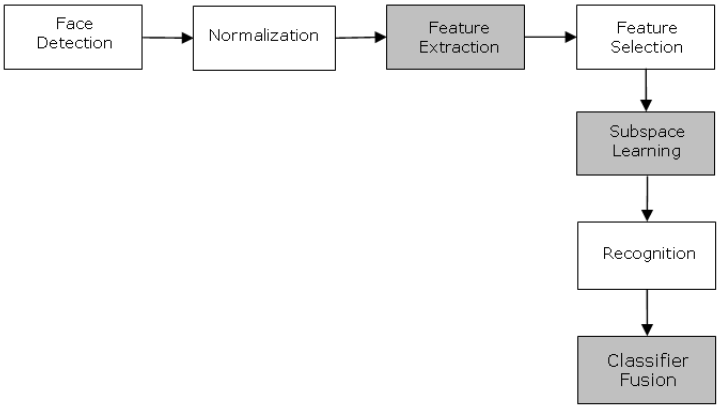

Figure 2.1: Face Recognition Pipeline

are many different methods for this single step. Details will be discussed in the following section.

- Feature Selection: if the length of the feature vector is too big, it might be beneficial to select a relevant subset of all features. This brings two good results. First, it improves the performance because of shorter feature vector. Second, selective features are more discriminant thus more accurate.

- Subspace Learning: this step is the main focus of the thesis. Its objective is to find a subspace into which all faces are projected. The subspace is normally learnt from training data. There are two main reasons to project the original data into lower-dimensional subspace. The first reason is that lower dimension leads to better performance. The second, and more important, reason is that the projected feature vectors in the subspace are believed to be more separable thus easier to recognise. In fact, this is the whole point of subspace learning.

- Recognition (identification or verification): the similarity between two faces is calculated using metrics like Euclidean distance, Cosine Similarity, or Mahalanobis

distance. In case of face identification, the class of the most similar face in training data is assigned to the testing face. In case of face verification, the similarity score is compared to a predefined threshold. Two faces belong to the same person if the score is bigger than the threshold, and they do not if otherwise.

- Classifier Fusion: there is a wide belief [11, 12] that the ensemble of multiple classifiers often outperforms the best classifier among them. The more diverse classifiers are, the better improvement the ensemble can make. In face images, there are many types of variations like lighting, scale, rotation, noise. Each feature extraction method can deal with a number of variations well but not all, and they are often complementary. For example, Local Binary Patterns is very good at dealing with local variation but not with global lighting change. On the other hand, Gabor Wavelets is invariant to scale, rotation and lighting but not specially designed to deal with local change. As expectedly, Local Binary Patterns and Gabor Wavelets are complementary to each other. The combination of these two types of features is called Local Gabor Binary Patterns (LGBP) which performs much better than each of them individually. LGBP is an example of combining features at feature level. In other words, features are combined at feature extraction step. There is another way to combine features which can be done after the recognition step. Multiple features produce multiple scores. These scores are combined using learning methods such as Logistic Regression, Support Vector Machines, and Evolutionary Algorithms.

In the following sections, we will focus the discussion on feature extraction, subspace learning and classifier fusion. These three topics are the most relevant to the contributions of the thesis. Comprehensive review on other topics can be found in [11, 13, 14,

15].

2.2 Feature Extraction

There are so many feature extraction methods used in face recognition. In this section, only the most popular ones are presented: Intensity values, Haar-like features, Gabor Wavelets, Local Binary Patterns, and Local Gabor Binary Patterns.

2.2.1 Intensity values

Face images are normally stored as 2D arrays in digital form. Each element in the array is called a pixel which has an intensity value. The simplest form of face representation uses the pixel intensities directly. All rows of the face image are combined to form a single vector called the feature vector. The advantage of this method is simplicity. However, because of its simplicity, it is unable to capture complex textures and handle the large variations which are generally found in human faces. Another limitation is that the feature vector is often in a very high dimensional space. Therefore, more efficient methods are desirable.

2.2.2 Haar-like features

Generally, Haar-like features are image features used in object recognition. In the context of face analysis, Haar-like features can be used for both face recognition [16] and face detection [2]. The name comes from its intuitive similarity with Haar Wavelets. As pointed out in the previous section, working with image intensities directly is com-

putationally expensive. In [17], Papageorgiou et al. proposed an alternate feature set instead of the usual intensity values. This feature set considers rectangular regions of the image and sums up the pixels in each region. This sum is used to categorise images. For example, let us say we have a set of images with buildings and human faces. If the eye and the hair region of the faces are considered, then it is very likely that the sum of the pixels in this region would be arbitrarily high or low for the buildings and quite high for the human faces. The value for the former would depend on the structure of the building and its environment while the values for the latter will be more or less the same. We could thus categorise all images with Haar-like feature in this rectangular region to be in a certain range of values as one category and those falling out of this range in another. This might roughly divide the set of images into ones having a lot of faces and a few buildings, and the other having a lot of buildings and a few faces. This procedure could be iteratively carried out to sub-divide the image clusters.

A simple way to define rectangular Haar-like feature is to calculate the difference of the sum of pixels of areas inside the rectangle, which can be at any position and scale within the original image. This modified feature set is called 2 rectangle feature. Viola and Jones [2] also defined 3 rectangle features and 4 rectangle features. The values indicate certain characteristics of a particular area of the image. Each feature type can indicate the existence (or not) of certain characteristics in the image, such as edges or changes in texture. For example, a 2 rectangle feature can indicate where the border is between a dark region and a light region. Four types of Haar-like features Viola and Jones used for face detection are illustrated in Figure 2.2. The sum of the pixels which lie within the white rectangles are subtracted from the sum of pixels in the grey rectangles.

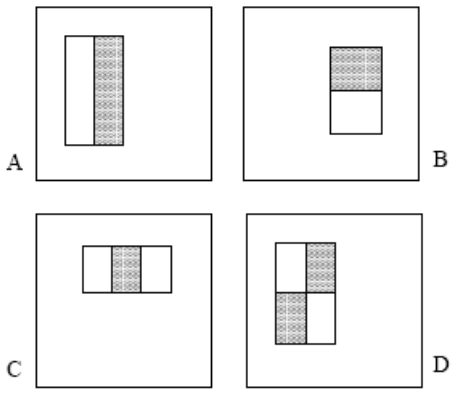

Figure 2.2: Haar-like features for face detection (*taken from* [2])

One important contribution of Viola and Jones was the use of Summed-area Tables, which they called Integral Images. Integral Images can be defined as 2-dimensional lookup table in the form of a matrix with the same size of the original image. Each element of the Integral Image contains the sum of all pixels located on the up-left region of the original image (in relation to the element's position). This allows the sum of rectangular areas in the image to be computed, at any position or scale, using only 4 lookups. For example, in Figure 2.3, the sum within D can be computed as:

$$D = pt_4 - pt_3 - pt_2 + pt_1$$

where pt_i is the sum of pixels within the rectangle from the top left corner to point i in the figure.

It might need more than 4 lookups to compute each Haar-like feature, depending on how it was defined. Viola and Jones's 2 rectangle features (type A, B) need 6 lookups, 3 rectangle features (type C) need 8 lookups, and 4 rectangle features (type D) need 9 lookups.

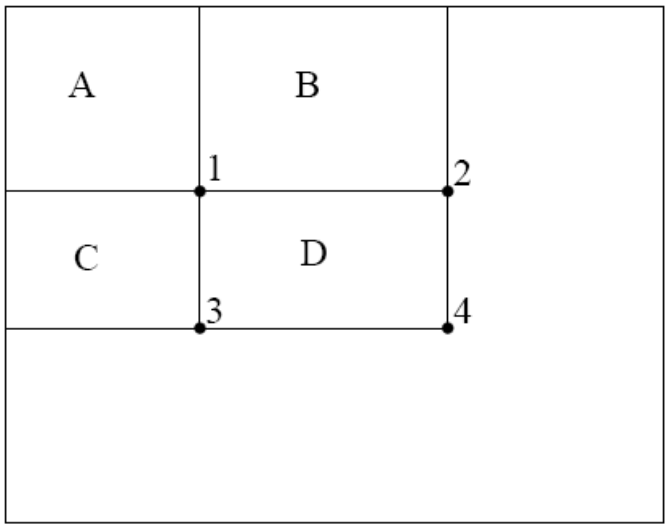

Figure 2.3: Integral Images

Haar-like features work very well in face detection. They not only achieve great detection rate, but also run at an extremely fast speed. Many successful face detection systems today evolved from Viola and Jones' ideas. Viola and Jones also applied their method to face recognition [16] but the result is not as impressive as the result for face detection. There are two main reasons. The first reason is that the variations in faces are sometimes too large for Haar-like features to handle well. The second reason is the lack of training data. Unlike in the case of face detection, face and non-face images are easily collected (there are a huge amount of images available online), training data for face recognition is much harder to collect and process.

In the next section, we will discuss a feature extraction method which is much more robust to variations, namely Gabor Wavelets.

2.2.3 Gabor Wavelets

In the spatial domain, the $2D$ Gabor filter is a Gaussian kernel function modulated by a sinusoidal plane wave [18]:

$$\varphi_{\Pi(f,\theta,\gamma,\eta)}(x,y) = \frac{f^2}{\pi\gamma\eta} e^{-\left(\alpha^2 x'^2 + \beta^2 y'^2\right)} e^{j2\pi f x'} \tag{2.2.1}$$

$$x' = x\cos\theta + y\sin\theta$$

$$y' = -x\sin\theta + y\cos\theta$$

where f is the central frequency of the sinusoidal plane wave, θ is the anti-clockwise rotation of the Gaussian and the plane wave, α is the sharpness of the Gaussian along the major axis parallel to the wave, and β is the sharpness of the Gaussian minor axis perpendicular to the wave. $\gamma = \frac{f}{\alpha}$ and $\eta = \frac{f}{\beta}$ are defined to keep the ratio between frequency and sharpness constant.

Image features can be extracted by convolving the image $I(x,y)$ with Gabor filters:

$$O_{\Pi(f,\theta,\gamma,\eta)}(x,y) = I * \varphi_{\Pi(f,\theta,\gamma,\eta)}(x,y) \tag{2.2.2}$$

Usually a number of Gabor filters of different scales and orientations are used. A filter bank with 5 scales and 8 orientations for feature extraction purposes is designed as:

$$f_u = \frac{f_{max}}{\sqrt{2}^u},\ \theta_v = \frac{v}{8}\pi,\ u = 0,\ldots,4,\ v = 0,\ldots,7$$

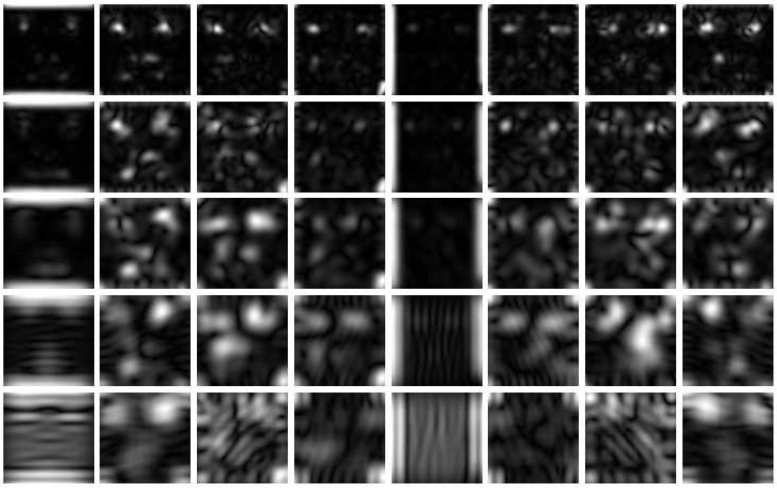

Figure 2.4: 40 Gabor Convolution Images

The resultant Gabor feature set thus consists of the convolution results of an input image I(x,y) with all of the 40 Gabor filters:

$$S = \{O_{u,v}(x,y) : u \in \{0,\ldots,4\}, v \in \{0,\ldots,7\}\} \tag{2.2.3}$$

where $O_{u,v}(x,y) = \left|I * \varphi_{\Pi(f,\theta,\gamma,\eta)}(x,y)\right|$, i.e. only the magnitude of the Gabor filter response is considered. Row vectors $O_{u,v}$ of $O_{u,v}(x,y)$ are then concatenated to generate a Gabor feature vector:

$$G(I) = O(I) = (O_{0,0}O_{0,1}\ldots O_{4,7}) \tag{2.2.4}$$

The number of Gabor filters used varies on a case by case basis. Usually, 40 filters (5 scales and 8 orientations) are used in face recognition applications. Figure 2.4 shows the convolution results of a face image with 40 Gabor filters. Thus an image of size 64×64 will give a feature vector of $64 \times 64 \times 5 \times 8 = 163,840$ dimensions.

Gabor filters were first introduced to face recognition by Lades et al. in [19]. They proposed the Dynamic Link Architecture (DLA), which recognises faces by extracting Gabor features at each node of a rectangular grid over the face image. In [20], Wiskott et al. extended DLA and proposed the famous Elastic Bunch Graph Matching (EBGM) method. The EBGM algorithm was the best performer in the FERET evaluation contest. However, both DLA and EBGM require extensive amount of computation - 30s on a SPARC station 10- 512 [20].

More recently, a simple but very powerful facial descriptor method called Local Binary Patterns (LBP) was also applied to face recognition in [21].

2.2.4 Local Binary Patterns

The original Local Binary Patterns (LBP) operator, introduced by Ojala et al. in [22], is a powerful method for texture description. The operator labels the pixels of an image by thresholding the 3×3-neighbourhood of each pixel with the centre value and representing the result as a binary number. The histogram of the labels can then be used as a texture descriptor. An illustration of the basic LBP operator is shown in Figure 2.5.

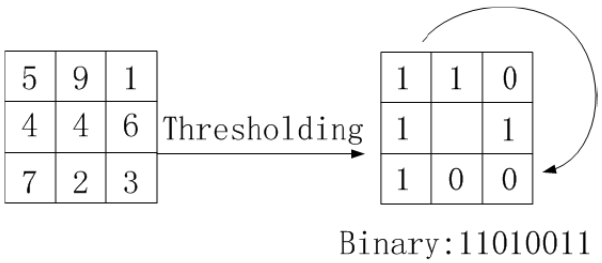

Figure 2.5: Local Binary Patterns (*taken from* [3])

An extension to the original operator is to use the uniform patterns [3]. A Local Bi-

nary Pattern is called uniform if it contains at most two bitwise transitions from 0 to 1 or vice versa when the binary string is considered circular. Ojala noticed that in their experiments with texture images, uniform patterns account for a little less than 90% of all patterns. As a result, only uniform patterns are often used for face recognition [23]. Unlike Gabor Wavelets, due to its simplicity, LBP is very fast to compute and complements Gabor Wavelets remarkably. In [24, 25], the authors combined LBP and Gabor Wavelets into a facial descriptor method called Local Gabor Binary Patterns (LGBP), which achieved excellent results on the FERET dataset. LGBP will be discussed in the next section.

2.2.5 Local Gabor Binary Patterns

Steps to extract Local Gabor Binary Patterns (LGBP) feature are illustrated in Figure 2.6 [24]. Specifically, a face image is modelled as a histogram sequence by the following steps:

1. An input face image is normalised and transformed to obtain 40 Gabor Magnitude Images (GMIs) in frequency domain by applying 5-scale and 8-orientation Gabor filters; GMIs can be generated by convolving the image with each of the 40 Gabor filters. For each Gabor filter, one magnitude value will be computed at each pixel position;

2. Each GMI is converted to Local Gabor Binary Pattern (LGBP) map;

3. Each LGBP Map is further divided into non-overlapping rectangular regions with specific size, and histogram is computed for each region; and

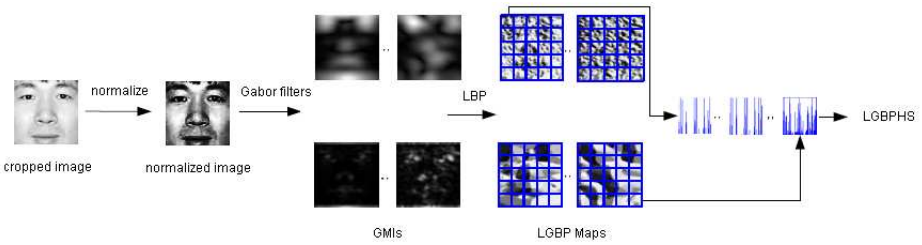

Figure 2.6: The proposed LGBPHS face representation approach.

4. The LGBP histograms of all the LGBP Maps are concatenated to form the final histogram sequence as the feature vector.

In step 2, the magnitude values with LBP operator are encoded to further enhance the information in the Gabor Magnitude Images (GMI). The original LBP operator [3] labels the pixels of an image by thresholding the 3×3 neighbourhood of each pixel $f_p(p = 0, 1, \ldots, 7)$ with the centre value f_c, as shown in Figure 2.5.

$$S(f_p - f_c) = \begin{cases} 1, & f_p \geqslant f_c \\ 0, & f_p < f_c \end{cases} \tag{2.2.5}$$

Then, by assigning a binomial factor 2^p to each $S(f_p - f_c)$, the LBP pattern at the pixel is

$$LBP = \sum_{p=0}^{7} S(f_p - f_c)2^p \tag{2.2.6}$$

which characterises the spatial structure of the local image texture. The operator LGBP denotes the LBP operates on GMI. $G_{lgbp}(x, y, \mu, \nu)$ denotes the transform result at position (x, y) of (μ, ν) - GMI, which composes the (μ, ν)- LGBP Map.

The histogram **h** of an image $f(x,y)$ with gray levels in the range [0, L-1] could be defined as

$$\mathbf{h_i} = \sum_{\mathbf{x,y}} I\left\{f(x,y) = i\right\}, i = 0, 1, \ldots, L-1 \tag{2.2.7}$$

where i is the i-th gray level, h_i is the number of pixels in the image with gray level i and

$$I\{A\} = \begin{cases} 1, & A\ is\ true \\ 0, & A\ is\ false \end{cases} \tag{2.2.8}$$

Assume each LGBP Map is divided into m regions $R_o, R_1, \ldots, R_{m-1}$. The histogram of the r-th region of the specific LGBP Map (from (μ,ν)-GMI) is computed by

$$H_{\mu,\nu,r} = (h_{\mu,\nu,r,0}, h_{\mu,\nu,r,1}, \ldots, h_{\mu,\nu,r,L-1}) \tag{2.2.9}$$

where

$$h_{\mu,\nu,r,i} = \sum_{(x,y)\in R_r} I\left\{G_{lgbp}(x,y,\mu,\nu) = i\right\} \tag{2.2.10}$$

Finally, all the histogram pieces computed from the regions of all the 40 LGBP Map are concatenated to a histogram sequence, R, as the final face representation $R = (H_{0,0,0}, \ldots, H_{0,0,m-1}, H_{0,1,0}, \ldots, H_{0,1,m-1}, \ldots, H_{4,7,m-1})$.

LGBP has many good properties inherited from both Gabor Wavelet and Local Binary Patterns: robust to noise, illumination, and local change. It performs really well on the FERET dataset [24, 26]. The downside is that the feature vector is in very high dimensional space. This leads to two problems: the feature extraction step is slow and the "curse of dimensionality" problem (Section 1.3). With Gabor Wavelets, it is possible to improve the speed of feature extraction step by only convolving a small

number of pixels with 40 Gabor filters. These pixels can be selected by simple subsampling method [27] or by more advanced AdaBoost learning [28]. Unfortunately, the same idea cannot be applied to Local Gabor Binary Patterns because Gabor filter convolution step is followed by Local Binary Patterns which requires the values of the entire image. In other words, there is no way to improve the speed of Local Gabor Binary Patterns. What can be done is to solve the "curse of dimensionality" problem.

After the feature vector is extracted, it is often in high dimensional space, thus contains redundant information and is slow to process. It is desired to reduce the dimension of feature vector while preserving as much useful information as possible. Methods designed to achieve that objective will be discussed next.

2.3 Subspace Learning

In Pattern Recognition and Machine Learning, there are three closely related learning techniques: subspace learning, metric learning, and dimension reduction. They are all designed to find a transformation of the original data so that the problem in hand is easier to solve with the transformed data. There are overlaps between these techniques. Their relationship is illustrated in Figure 2.7. Nonlinear dimension reduction is also known as manifold learning. As shown in the diagram, manifold learning is not the same as subspace learning since a nonlinear manifold can not be represented by a subspace. However, because of a lack of a general term, all of these methods will be called under the same name "subspace learning" in this thesis.

One important point to make here is that terms "metric learning" and "subspace learning" are used interchangeably throughout the thesis. More strictly speaking, metric

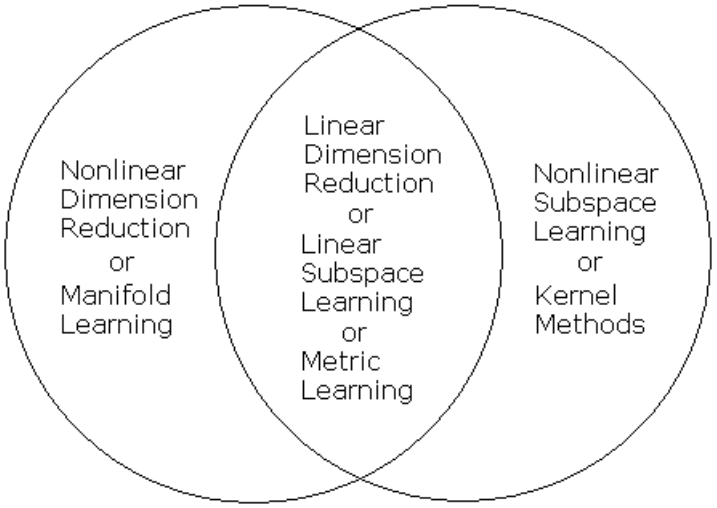

Figure 2.7: The Relationship between Subspace Learning, Metric Learning, and Dimension Reduction

learning is equivalent to linear subspace learning. Hence, the relationship between metric learning and linear subspace learning requires an explanation which is as follows.

The objective of linear subspace learning is to learn a linear transformation matrix A from a training data set. On the other hand, the objective of metric learning methods is to learn a distance metric from a training data set so that under that metric, data points within the same class are close and data points from different classes are far way. More formally, in metric learning, the distance between points $x \in R^m$ and $y \in R^m$ is defined as

$$d(x, y) = (x - y)^T Q(x - y) \tag{2.3.1}$$

The typical problem of metric learning is the learning of the $Q \in R^{m \times m}$. Eq (2.3.1) can be further written as

$$\begin{aligned} d(x,y) &= (x-y)^T Q^{\frac{1}{2}^T} Q^{\frac{1}{2}} (x-y) \\ &= (Q^{\frac{1}{2}}x - Q^{\frac{1}{2}}y)^T (Q^{\frac{1}{2}}x - Q^{\frac{1}{2}}y) \\ &= (Ax-Ay)^T (Ax-Ay) \end{aligned}$$

where $A = Q^{\frac{1}{2}}$. It is clear that the learning of Q is equivalent to the learning of a linear transformation matrix A in the original space. Therefore, metric learning and linear subspace learning are effectively equivalent.

This chapter does not review all subspace learning methods but only those which are most relevant to face recognition. The following are methods discussed in each category.

- Nonlinear Dimension Reduction: Isomap, and Locally Linear Embedding.
- Linear Subspace Learning (or Linear Dimension Reduction): Principle Component Analysis, Multi-Dimensional Scaling, Linear Discriminant Analysis, Neighbourhood Component Analysis, Large Margin Nearest Neighbour, and Information-Theoretic Metric Learning.
- Kernel Methods: Kernel PCA, and Kernel LDA.

Because of the overlaps between three learning techniques, it is difficult to discuss methods under two categories of linear and nonlinear. Instead, they are divided into two categories in a different way: Unsupervised Learning and Supervised Learning.

2.3.1 Unsupervised Learning

The objective of unsupervised learning is to discover the underlying structure in unlabelled training data. In context of face recognition, it usually means finding a transformation of the original data to lower-dimensional subspace. The transformation can be linear or nonlinear. In this section, we will discuss two linear and two nonlinear methods which are popular in the field of face recognition: Principal Component Analysis, Multi-Dimensional Scaling, Isomap, and Locally Linear Embedding.

2.3.1.1 Principle Component Analysis

The aim of Principle Component Analysis (PCA) is to identify a subspace spanned by the training images $\{x_1, x_2, \ldots, x_n\}$, which could decorrelate the variance of pixel values. This can be achieved by eigen analysis of the covariance matrix $\Sigma = \frac{1}{n}\sum_{i=1}^{n} \Phi_i \Phi_i^T$:

$$\Sigma E = \Lambda E \tag{2.3.2}$$

where E, Λ are the resultant eigenvectors, also referred to as eigenfaces, and eigenvalues respectively. The representation of a face image in the PCA subspace is then obtained by projecting it into the coordinate system defined by the eigenfaces [29].

PCA is simple thus easy to understand. It involves single eigen decomposition step which is therefore fast to compute. Moreover, PCA is very good for dimension reduction because it preserves a large amount of information in a small number of dimensions. The limitation is that PCA is not designed to solve the classification problem. While it maximises the variations in projected dimensions, it is unclear these variations represent the difference of identities or just the difference of the same person.

In chapter 3, however, it will be shown that if used with whitening process, PCA can perform really well in face recognition, even when there is a lack of training data.

Another limitation of PCA is it is based on the assumption that data points are lying in a linear subspace, which might not be the case of face images. Faces images are believed to lie in a nonlinear manifold. That is why nonlinear methods will be discussed. Before moving to nonlinear methods, let us discuss another popular linear method, Multi-Dimensional Scaling.

2.3.1.2 Multi-Dimensional Scaling

Multi-Dimensional Scaling (MDS) is also used to detect underlying structure of training data. Unlike PCA, MDS tries to preserve the distances between data points as much as possible. MDS starts with a matrix of item-item similarities (dissimilarities) and then assigns a location to each item in d-dimensional space, where d is specified as a priori. For the purpose of dimension reduction, d is often chosen to be small. Formally, assuming that there is a set of N objects, the dissimilarity (distance) matrix $N \times N$ is defined as:

$$\Delta = \begin{bmatrix} \delta_{1,1} & \delta_{1,2} & \cdots & \delta_{1,N} \\ \delta_{2,1} & \delta_{2,2} & \cdots & \delta_{N,2} \\ \vdots & \vdots & & \vdots \\ \delta_{N,1} & \delta_{N,2} & \cdots & \delta_{N,N} \end{bmatrix}$$

where $\delta_{i,j}$ is the distance between object i and object j. The objective of MDS is to find N vector $x_1, x_2, \ldots, x_N \in R^d$ such that

$$\left\|x_i - x_j\right\| \approx \delta_{i,j} \; for \; all \; (i, j)$$

where $\|.\|$ is a vector norm. Generally, this norm may be an an arbitrary distance function but in classical MDS, it is often the Euclidean distance. Usually, MDS is formulated as an optimisation problem, where $(x_1, x_2, \ldots, x_N)$ is found as a minimiser of some cost function, for example,

$$min_{x_1,x_2,\ldots,x_N} \sum_{i<j} (\left\|x_i - x_j\right\| - \delta_{i,j})^2$$

The solution to this problem can be found using numerical optimisation techniques. Although MDS is not often used in face recognition directly, its nonlinear extension, Isomap, is quite popular.

2.3.1.3 Isomap

Isomap is a globally nonlinear manifold learning method invented by Joshua et. al. in 2000 [30]. It is a nonlinear extension of Multi-Dimensional Scaling (MDS) technique. The only difference between Isomap and MDS is that Isomap tries to preserve the geodesic manifold distances instead of normal distances. The key idea is estimating the geodesic distance between faraway points, given only input-space distances. Intuitively, input space distance provides a good approximation to geodesic distance for neighbouring points. For faraway points, geodesic distance can be approximated by summing up a sequence of short distances between neighbouring points. By finding shortest paths in a graph with edges connecting neighbouring data points, we can compute these approximations efficiently. The entire algorithm has three steps:

1. Step one determines which points are neighbours on the manifold M, based on the distances $d_X(i,j)$ between pairs of points (i,j) in the input space X. There are two simple methods: to connect to all of its K nearest neighbours, or to each point to all points within some fixed radius ϵ. These neighbourhood connections form a weighted graph G over the data points, with edges of weight $d_X(i,j)$ between neighbouring points.

2. In the second step, the geodesic distances $d_M(i,j)$ between all pairs of points on the manifold M are estimated by computing their shortest path distances $d_G(i,j)$ in the graph G.

3. The final step to construct an embedding of the data in a d-dimensional Euclidean space Y that best preserves the manifold's estimated intrinsic geometry. It can be done by applying classical MDS to the matrix of graph distances $d_G = \{d_G(i,j)\}$,

In the context of face recognition, Isomap can be used to reduce the dimension of feature vector while preserving important information of the data (like PCA). Isomap can also be used to select a number of representative frames in a set of frames [31].

Another nonlinear manifold learning algorithm will be discussed next.

2.3.1.4 Locally Linear Embedding

Locally Linear Embedding (LLE) was introduced in [32] by Sam et. al. at approximately the same time as Isomap. It has several advantages over Isomap, including better results with many problems, and faster optimisation when implemented to take advantage of sparse matrix algorithms. Similar to Isomap, LLE begins by finding a set

of the nearest neighbours of each point. Then a set of weights for each point is calculated to best describe the point as a linear combination of its neighbours. Finally, LLE uses an eigenvector-based optimisation technique to find the low-dimensional embedding of points, such that each point is still described with the same linear combination of its neighbours. As there is no fixed unit to prevent the weights from drifting as various regions differ in sample densities, LLE tends to handle non-uniform sample densities poorly.

The barycentric coordinates of a point X_i can be computed from its neighbours X_j. The original point is reconstructed by a linear combination, given by the weight matrix W_{ij} of its neighbours. The reconstruction error is given by the cost function $E(W)$.

$$E(W) = \sum_i \left| X_i - \sum_j W_{ij} X_j \right|^2$$

The weights W_{ij} represent the amount of contribution the point X_j has while reconstructing the point X_i. The cost function $E(W)$ is minimised under 2 constraints: (a) The sum of every row of the weight matrix equals 1, i.e. $\sum_j W_{ij} = 1$ and (b) Each data point X_i is reconstructed only from its neighbours, i.e. if point X_j is not a neighbour of the point X_i, W_{ij} is enforced to be zero.

LLE's objective is to reduce the dimension of the original data points, D, to a smaller dimension, d such that $D>>d$. The same weights W_{ij} reconstructing the ith data point in the D-dimensional space will be used to reconstruct the same point in the lower d-dimensional space. A neighbourhood preserving map is created based on this idea. Each point X_i in the D-dimensional space is mapped onto a point Y_i in the d-dimensional space by minimizing the cost function.

$$C(Y) = \sum_i \left| Y_i - \sum_j W_{ij} Y_j \right|^2$$

Unlike in $E(W)$, the weights W_{ij} in $C(Y)$ are kept fixed and the minimisation is done on the points Y_i to optimise the coordinates. This minimisation problem can be solved by solving a sparse $N \times N$ eigenvalue problem, whose bottom d nonzero eigenvectors provide an orthogonal set of coordinates. In general, the data points are reconstructed from K nearest neighbours, as measured by Euclidean distance. In such an implementation, K is the only one free parameter and can be estimated by cross validation.

The key assumption in the LEE algorithm is that each data point can be approximated by linear combination of neighbour data points. Therefore, LLE is often used to process video sequences instead of static images. For example, in chapter 7, LLE is used to select a small number of most representative frames in a video [31]. In the context of face recognition, LLE has the same uses as Isomap. The advantage of LLE over Isomap is the speed.

In this section, four popular unsupervised learning methods: Principal Component Analysis, Multi-Dimensional Scaling, Isomap, and Locally Linear Embedding have been presented. They all try to discover the underlying structure of unlabelled training data. However, if labelled data samples are provided then the additional information can be made use of using another type of learning technique called supervised learning.

2.3.2 Supervised Learning

If labelled samples are provided, supervised learning techniques are often more suitable to solve the problem of face recognition because they are specifically designed for classification. In a number of cases, however, supervised methods do not perform well as expected because of a problem called "overfitting". Overfitting occurs when the statistical model describes random error instead of the underlying structure of data. The problem becomes greater if there is a lack of training data. In those cases, additional care needs to be taken in order to avoid or reduce the effect of overfitting. Next supervised learning techniques most used in face recognition will be discussed together with mechanisms to limit overfitting.

2.3.2.1 Linear Discriminant Analysis

Linear Discriminant Analysis (LDA) is one of the first supervised learning technique invented. Different from PCA, LDA is aimed at finding a projection matrix W which maximises the quotient of determinants of the between-class scatter S_b and the within-class scatter S_w [33]. W is defined by the following formula:

$$W = argmax \frac{|W^T S_b W|}{|W^T S_w W|} \tag{2.3.3}$$

Consider a C-class classification problem and let N_c be the number of samples in class c, a set of n training samples from C classes can be defined as

$$\{x_{ik}, i = 1, 2, \ldots, C; k = 1, 2, \ldots, N_c\}, n = \sum_{i=1}^{C} N_i$$

Let μ be the mean of the whole training set and μ_c be the mean for class c, S_w and S_b of

a training set can be computed as follows:

$$S_w = \frac{1}{C}\sum_{c=1}^{C}\frac{1}{N_c}\sum_{k=1}^{N_c}(x_{ck}-\mu_c)(x_{ck}-\mu_c)^T \tag{2.3.4}$$

$$S_b = \frac{1}{C}\sum_{c=1}^{C}(\mu_c-\mu)(\mu_c-\mu)^T \tag{2.3.5}$$

According to [34], the projection matrix W can be computed from the eigenvectors of $S_w^{-1}S_b$. The high dimensionality of the feature vector, however, usually makes S_w singular, especially in face recognition applications. This means the inverse of S_w does not exist., resulting in a two-stage dimensionality reduction technique called the Most Discrimiant Features (MFD) [35]. Accordingly, PCA is first used to project the original face vectors to a lower-dimensional space and then LDA is used with the transformed data. Let W_{pca} be the projection matrix from the original image space to the PCA subspace, the LDA projection matrix W_{lda} is thus composed of the eigenvectors of $(W_{pca}^T S_w W_{pca})^{-1}\left(W_{pca}^T S_b W_{pca}\right)$. The final projection matrix W_{mfd} is accordingly calculated as:

$$W_{mfd} = W_{pca}W_{lda} \tag{2.3.6}$$

It is noted here that the rank of $S_b \leq C-1$ and the rank of $S_w \leq M-C$. Consequently, the dimension of the PCA subspace should be $M-C$ [35].

Although MFD solves the singularity problem of S_w, it overfits the training data thus lacks the generalisation ability. In the PCA step, relevant questions concerning PCA are usually related to the range of Principal Components (PCs) used and how it affects performance. Regarding discriminant analysis, it is essential to understand the causes

of overfitting and how to avoid it. The answers to these two questions are closely related. One can actually show that using more PCs may lead to decreased performance (for recognition). One possible explanation is that the trailing eigenvalues correspond to high-frequency components and often encode noise. Therefore, the FLD procedure has to fit for noise after these trailing but small valued eigenvalues are used to define the reduced PCA subspace. This in turn causes the overfitting effect.

In [36], Chengjun Liu et al. proposed two methods to improve the generalisation ability of FLD called Enhanced Fisher Linear Discriminant Model (EFM) 1 and 2. The objective of EFM-1 is to balance between the need that the selected eigenvalues account for most of the spectral energy of the raw data and the requirement that the eigenvalues of the within-class scatter matrix in the reduced PCA subspace are not too small. EFM-2 implements the dimensionality reduction as Fisherfaces does. Then it applies the whitening process to the within-class scatter matrix in the reduced PCA subspace. After that, it chooses a small set of features (corresponding to the eigenvectors of the within-class scatter matrix) so that the smaller trailing eigenvalues are not included in further computation of the between-class scatter matrix. These two improved methods, EFM 1 and 2, do improve the generalisation ability of LDA but the accuracy improvement is not significant enough [36]. Therefore, better ways to increase the generalisation ability of LDA is still needed.

Most classical subspace methods like PCA or LDA use very simple form of objective functions which can be optimised with eigen-decomposition. While PCA tries to maximise the projected variations, LDA tries to maximise the quotient of determinants of the between-class scatter S_b and the within-class scatter S_w. Recently, many methods explored the possibility of more advanced objective functions and achieved excellent

results [9, 4, 37, 38]. These methods try to find a linear transformation from the original space to the linear subspace with the objective of optimizing some objective function. A recently invented subspace method called Neighbourhood Component Analysis will be discussed in the next section.

2.3.2.2 Neighbourhood Component Analysis

In 2005, Goldberger et al. [9] proposed Neighbourhood Component Analysis (NCA), a distance metric learning algorithm especially designed to improve kNN classification. The idea is to learn a Mahalanobis distance by minimizing the leave-one-out cross validation error of the kNN classifier on a training set. Specifically, given a labelled data set consisting of n real-valued input vectors $x_1, x_2, ..., x_n \in R^D$ and corresponding class labels $c_1, c_2, ..., c_n$, the objective is to find a distance metric that maximises the performance of nearest neighbour classification. Because the performance on future test data is unknown, NCA attempts to optimise the leave-one-out (LOO) performance on the training data. Learning a Mahalanobis distance metric is equivalent to learning a linear transformation of the input space. If the transformation is denoted as a matrix A, we are effectively learning a metric $Q = A^T A$ such that $d(x, y) = (x - y)^T Q(x - y) = (Ax - Ay)^T (Ax - Ay)$. The objective function in NCA is based on stochastic neighbour assignments in the transformed space. In particular, each point i selects another point j as its neighbour with some probability p_{ij}, and inherits its class label from the point it selects. p_{ij} is defined as follows:

$$p_{ij} = \frac{e^{-\left\|Ax_i - Ax_j\right\|^2}}{\sum_{k \neq i} e^{-\|Ax_i - Ax_k\|^2}}, \quad (p_{ii} = 0)$$

Under this stochastic selection rule, the probability that point i will be correctly classified can be computed as (denote the set of points in the same class as i by $C_i = \{j|c_i = c_j\}$):

$$p_i = \sum_{j \in C_i} p_{ij}$$

Then the objective function being maximised is the expected number of points correctly classified under this scheme:

$$f(A) = \sum_i \sum_{j \in C_i} p_{ij} = \sum_i p_i$$

Since the objective function is differentiable with regard to matrix A, it can be optimised by using a gradient-based optimiser such as delta-bar-delta or conjugate gradients (details can be found in [9]). NCA has a number of advantages over traditional methods like PCA or LDA. First, being a non-parametric method, it does not need to make any assumption about data distribution. Both PCA and LDA assume Gaussian distribution of data. Second, the objective function in NCA is designed to directly minimise the cross validation error while PCA and LDA try to optimise indirect criteria which are the variation and the quotient of determinants of the between-class scatter and the within-class scatter. Third, through experiments, NCA is shown to be less sensitive to the overfitting problem than LDA [9]. NCA, however, also has several limitations. The first limitation is that the objective function is non-convex so there is no guarantee of finding globally optimal solution. The second limitation is that the objective function is rather complex thus very time consuming to compute its gradient. As a result, NCA cannot deal with high dimensional data well such as in the case of face recognition. The

usual solution is to apply PCA first to reduce the dimension to a manageable number. Next we will discuss another metric learning method which also try to optimise the kNN classification performance.

2.3.2.3 Large Margin Nearest Neighbour

In 2006, Weinberger et al. [4] proposed a method called Large Margin Nearest Neighbour (LMNN) that learns a matrix designed to improve the performance of kNN classification using semidefinite programming. Similar to Neighbourhood Component Analysis, LMNN also looks for a transformation matrix to optimise the kNN performance in the subspace. In LMNN, the objective function is composed of two terms. The first term minimises the distance between target neighbours. The second term is a hinge-loss that encourages target neighbours to be at least one distance unit closer than points from other classes. The general idea is to keep the k-nearest neighbours belong to the same class while examples from different classes are separated by a large margin (details can be found in [4]).

Figure 2.8 illustrates an example of one input's neighbourhood $\vec{x}_i$ before training versus after training. The distance metric is optimised so that (i) its $k = 3$ target neighbours lie within a smaller radius after training; (ii) differently labelled inputs lie outside this smaller radius, with a margin of at least one unit distance. Arrows indicate the gradients on distances arising from the optimisation of the objective function.

One advantage of LMNN over NCA is that its objective function is convex thus can be optimised using convex optimisation which guarantees to find the globally optimal solution. LMNN performs very well on a number of standard datasets including the

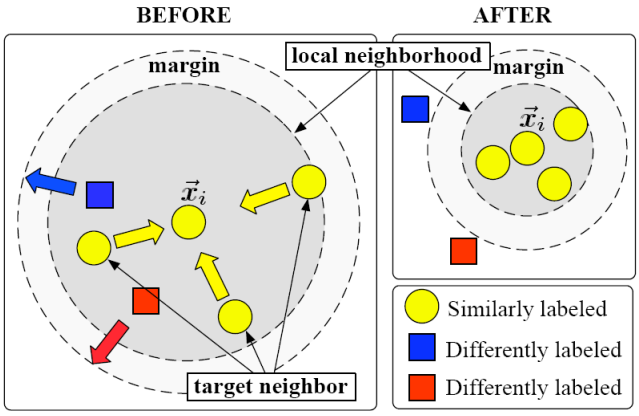

Figure 2.8: Illustration of LMNN with three classes before and after the transformation (*taken from* [4])

ORL face dataset [4]. Like NCA, however, LMNN requires information about the class of each sample. There is another state-of-the-art metric learning method which does not require the class label of each sample.

2.3.2.4 Information-Theoretic Metric Learning

Davis et al. [37] have taken an information theoretic approach (ITML) to learn a Mahalanobis metric under a wide range of possible constraints and prior knowledge on the Mahalanobis distance. Similar to NCA, ITML tries to find a positive definite matrix $Q = A^T A$ which parameterises the (squared) Mahalanobis distance:

$$d_Q(x_i, x_j) = (x_i - x_j)^T Q (x_i - x_j)$$

so that all constraints are satisfied. Typically, the constraints will be of the form $d_Q(x_i, x_j) \leq u$ for positive pairs and $d_Q(x_i, x_j) \geq l$ for negative pairs. In many cases, prior knowledge about the Mahalanobis distance function itself is known. For instances, if data is

Gaussian then parameterising the distance function by the inverse of the sample covariance matrix may be appropriate; or squared Euclidean distance may work well empirically. Thus, ITML regularises the Mahalanobis matrix Q to be as close as possible to a given Mahalanobis distance function, parameterised by Q_0. The closeness is measured as a Kullback-Leibler divergence between two Gaussian distributions corresponding to the two matrices Q and Q_0. Formally, given pairs of similar points S and pairs of dissimilar points D, the distance metric learning problem is

$$\begin{aligned} min_Q \quad & KL(p(x;Q_0)||p(x;Q)) \\ subject\ to \quad & d_Q(x_i,x_j) \leq u \quad (i,j) \in S \\ & d_Q(x_i,x_j) \geq l \quad (i,j) \in D \end{aligned}$$

where $KL(p(x;Q_0)||p(x;Q))$ is the differential relative entropy between the corresponding multivariate Gaussians of Q_0 and Q. Details of how to solve this optimisation problem can be found in [37]. ITML has two big advantages over other metric learning methods. First, it can handle a wide variety of constraints and can optionally incorporate a prior on the distance function. Second, it is fast and scalable. Because of these properties, ITML is applicable to a wide range of applications. For example, ITML produces state-of-the-art results on many UCI datasets [37].

Since ITML uses the constraints under the form of interpoint similarity, it is more suitable for face verification problem. In [38], Matthieu et al. applied ITML to face verification on the state-of-the-art dataset, Labelled Faces in the Wild, and achieved excellent results.

Despite the great success of linear subspace methods in face recognition, many researchers believe that faces lie in a nonlinear manifold thus nonlinear methods are more

suitable to solve the problem. The main idea of creating nonlinear version from a linear method will be briefly discussed next.

2.3.3 Kernel methods

As discussed in the last section, linear methods only search for linear subspace. Since facial variations are mostly nonlinear, these linear methods are believed to only provide sub-optimal solutions to face recognition tasks [39]. Recently, kernel methods have been successfully applied to pattern recognition because of their ability to handle nonlinear data. For example, Support Vector Machines (SVMs) are typical kernel methods and have been successfully applied to face detection [40], face recognition [41], and gender classification [42]. The key idea is that by mapping the input data to a higher dimensional feature space, a nonlinear problem defined in the original space is effectively turned into a linear problem in the feature space [43]. Linear methods can subsequently be performed in the feature space. Using this kernel trick, we can convert linear methods to their nonlinear versions such as Kernel Principal Component Analysis (KPCA) [44] converted from PCA, or Kernel Discriminant Analysis (KDA) [45] converted from LDA. Experiments show that KPCA and KDA are able to extract nonlinear features, thus provide better recognition rates in applications such as character [44] and face recognition [46]. How to convert PCA to its nonlinear counterpart, KPCA, will be discussed next (similarly LDA can be converted to KDA).

2.3.3.1 Kernel Principal Component Analysis

Suppose there are N training samples $\{x_i \in R^D, i = 1, 2, \ldots, N\}$ in the input space and ϕ is the nonlinear mapping defined from the input space to a high dimensional feature space: $\phi : R^D \rightarrow F$. Then each vector x_i can be mapped to a higher dimension vector $\phi(x_i)$ in the feature space. For the time being, let us assume that all the data mapped into the feature space are centred, i.e.,

$$\sum_{i=1}^{N} \phi(x_i) = 0 \tag{2.3.7}$$

The covariance matrix of the training samples in the feature space can be calculated as:

$$C = \frac{1}{N} \sum_{i=1}^{N} \phi(x_i)\phi(x_i)^T \tag{2.3.8}$$

Similar to PCA, Kernel PCA aims to find the eigenvalues $\lambda \geqslant 0$ and eigenvectors $v \in F \backslash \{0\}$ satisfying

$$\lambda v = Cv \tag{2.3.9}$$

All solutions v lie in the span of $\phi(x_1), \ldots, \phi(x_N)$, and there exist N coefficients $\alpha_1, \ldots, \alpha_N$ such that

$$v = \sum_{i=1}^{N} \alpha_i \phi(x_i) \tag{2.3.10}$$

By taking the inner-product with vector $\phi(x_i)$ $(i = 1, \ldots, N)$ on both sides of Eq (2.3.9), we obtain:

$$\lambda\left(v.\phi(x_i)\right) = (Cv).\phi(x_i) \tag{2.3.11}$$

By substituting Eq (2.3.8) and Eq (2.3.10) into Eq (2.3.11) and defining a $N \times N$ matrix $\mathbf{K}$ with:

$$\mathbf{K}_{ij} = k(x_i, x_j) = \phi(x_i)\phi(x_j) \tag{2.3.12}$$

we can have the following:

$$N\lambda\mathbf{K}\alpha = \mathbf{K}^2\alpha \Rightarrow N\lambda\alpha = \mathbf{K}\alpha \tag{2.3.13}$$

where α denotes a column vector with entries $\alpha_1, \ldots, \alpha_N$.

For a new pattern x, the projection of its image $\phi(x)$ in the feature space onto the eigenvector v can now be computed as:

$$v.\phi(x) = \sum_{i=1}^{N} \alpha_i \left(\phi(x_i).\phi\left(x\right)\right) = \sum_{i=1}^{N} \alpha_i k(x_i, x) \tag{2.3.14}$$

If the first L $(1 \leqslant L \leqslant N)$ significant eigenvectors are extracted to construct the eigen matrix:

$$\mathbf{W} = [\alpha_1\alpha_2 \ldots \alpha_L] \tag{2.3.15}$$

The projection of x in the L-dimensional Kernel PCA space is given by:

$$y = k_x\mathbf{W} \tag{2.3.16}$$

where $k_x = [k(x, x_1)k(x, x_2) \dots k(x, x_N)]$.

All the data mapped into the feature space have been assumed to be acentred. This is often not the case, therefore the following step is needed.

2.3.3.2 Non-centred Data

Generally, data $\{\phi(x_i)\}, i = 1, 2, \dots, N$ is not centred in the feature space. The following technique can be used to make this data mean zero:

$$\Phi(x_i) = \phi(x_i) - \frac{1}{N}\sum_{p=1}^{N}\phi(x_p) \tag{2.3.17}$$

Then the $N \times N$ kernel matrix $\mathbf{K}'$ for the centred data can be calculated as:

$$\left(\mathbf{K}'\right)_{ij} = \Phi(x_i)\Phi(x_j)$$

$$= \left(\phi(x_i) - \frac{1}{N}\sum_{p=1}^{N}\phi(x_p)\right).\left(\phi(x_j) - \frac{1}{N}\sum_{q=1}^{N}\phi(x_q)\right)$$

$$= \left(\phi(x_i).\phi(x_j)\right) - \frac{1}{N}\sum_{p=1}^{N}\phi(x_p).\phi(x_j) - \frac{1}{N}\sum_{q=1}^{N}\phi(x_i).\phi(x_q) + \frac{1}{N^2}\sum_{p=1}^{N}\sum_{q=1}^{N}\phi(x_p)\phi(x_q)$$

$$= (\mathbf{K})_{ij} - \frac{1}{N}\sum_{p=1}^{N}(\mathbf{K})_{pj} - \frac{1}{N}\sum_{q=1}^{N}(\mathbf{K})_{iq} + \frac{1}{N^2}\sum_{p=1}^{N}\sum_{q=1}^{N}(\mathbf{K})_{pq} \tag{2.3.18}$$

$\mathbf{K}'$ can also be presented in matrix form as follows:

$$\mathbf{K}' = \mathbf{K} - \frac{1}{N}\mathbf{1}_N\mathbf{K} - \frac{1}{N}\mathbf{K}\mathbf{1}_N + \frac{1}{N^2}\mathbf{1}_N\mathbf{K}\mathbf{1}_N \tag{2.3.19}$$

where $\mathbf{1}_N$ is the column vector, i.e. $N \times 1$ matrix, with all entries filled with 1.

Once the kernel matrix $\mathbf{K}'$ for the centred data is calculated, the same procedure as used in the previous section can be used to compute the projection matrix W for the KPCA subspace. As given in Equation 2.3.16, the projection of a new pattern x into the learned subspace can now be computed as:

$$y = k'_x \mathbf{W} \tag{2.3.20}$$

where

$$\left(k'_x\right)_i = \Phi(x)\Phi(x_i)$$

$$= \left(\phi(x) - \frac{1}{N}\sum_{p=1}^{N}\phi(x_p)\right) . \left(\phi(x_i) - \frac{1}{N}\sum_{q=1}^{N}\phi(x_q)\right)$$

$$= k(x, x_i) - \frac{1}{N}\sum_{p=1}^{N}k(x_i, x_p) - \frac{1}{N}\sum_{q=1}^{N}k(x, x_q) + \frac{1}{N^2}\sum_{p=1}^{N}\sum_{q=1}^{N}k(x_p, x_q) \tag{2.3.21}$$

We define a $1 \times N$ row vector $\mathbf{1}$ with all entries filled with 1, then the equation can be represented in a matrix form:

$$k'_x = k_x - \frac{1}{N}\mathbf{1}\mathbf{K} - \frac{1}{N}k_x\mathbf{1}_N + \frac{1}{N^2}\mathbf{1}\mathbf{K}\mathbf{1}_N \tag{2.3.22}$$

In this section, linear and nonlinear subspace learning methods have been discussed in a great detail. Each of them has its own advantages and limitations. The good news is that multiple methods can be combined to produce even better results.

2.4 Classifier Fusion

Whenever multiple information sources are available, it is important to have a method to combine all of them to produce a combined decision which is better than any individual decision. A multiple classifier system is a powerful solution to difficult pattern recognition problems since it allows simultaneous use of arbitrary feature descriptors and classification techniques, especially in case of noisy data. In this section, three techniques which are most relevant to the contributions of this thesis: voting scheme, logistic regression, and support vector machines will be discussed.

2.4.1 Voting scheme

Voting scheme is the simplest form of classifier combination. Voting works on class labels assigned to each pattern by the respective classifiers. The labels are acquired by hardening the soft decision outputs using the maximum value selector. The Vote rule output is a function of the votes collected for each class from each single classifier. A popular example of voting method is Borda count.

Borda count is a generalisation of the majority vote. The Borda count for a class is the total of the number of classes ranked below it by each classifier. Arranging the classes so that their Borda counts are in descending order, gives the consensus ranking.

The magnitude of the Borda count for each class measures the strength of agreement by the classifiers assumingly that the input pattern belongs to that class. For a binary problem, the Borda count is equal to the simple majority vote. Variations of the Borda count function, such as those which can handle ties in the rankings, are discussed in [47].

The Borda count function assumes that the contributions of the individual classifiers are independent. Using this method, a classifier is redundant if it always reinforces errors made by the others, that is, if all the classes which are ranked above a true class by that classifier are always contained in some other classifier's choices above the true class.

The Borda count method is simple to implement and requires no training. Despite its simplicity, Borda count achieved excellent results on the FERET face dataset [48]. However, it does not take into account the differences in the individual classifier capabilities. All classifiers are treated equally, which may not be preferable when certain classifiers are known to be more likely to be correct than others. Two methods aimed to address this limitation, namely Logistic Regression and Support Vector Machine, will be discussed next.

2.4.2 Logistic Regression

By assigning weights to the rank scores produced by each classifier, we can modify the Borda count method so that it can combine classifiers with non-uniform performances. The weights should reflect the relative significance of each classifier estimated in the context of the combination. Additionally, it will be useful to measure the confidence of the combined decisions given by the Borda count method. Possible measures can be a statistic derived from the distribution of sums of a given range of ranks, or the intervals between the computed Borda counts for a given set of classes [49].

The distribution of rank totals is affected by the correlation between the classifiers. However, that distribution does not necessarily imply classification correctness. For

instance, if rankings by two identical classifiers are combined, the rank sum for the class that is their common top choice falls on an extreme of the distribution, whether that decision is correct or not. The two effects, namely classifier correlation and classification correctness, must be differentiated and modelled separately.

Motivated by the need to differentiate the correct classes from the incorrect ones, we associate a binary variable Y_c to each class c for a given pattern. Y_c has the value 1 if c is the true class of that pattern, and 0 otherwise. The objective of recognition is therefore to predict the value of Y_c for each class c. Hence the decision combination problem can be reformulated in the context of regression analysis. The rank scores produced by each classifier are considered as random variables that are used to predict the value of Y_c for each class c, and their effects on Y_c can be modelled by a multiple regression function. Since Y_c is binary, a logistic response function is useful in this context [50, 51].

For simplicity, we denote the response variable Y_c by Y, which has a value for each class with respect to each input pattern: $Y = 1$ for the true class and $Y = 0$ for other classes. For a training pattern, the true class is known and therefore each class has a known value of Y. For an unseen pattern, the value of Y for each class has two possible outcomes.

Denoting the probability $P(Y = 1 \mid x)$ by $\pi(x)$, where $x = (x_1, x_2, \ldots, x_m)$, represents the rank scores assigned to that class by classifiers $C_1, C_2, \ldots, C_m$. For convenience in discussion, we assume that x_i has the largest value if the class is ranked at the top by C_i. Using the logistic response function

$$\pi(x) = \frac{exp\,(\alpha + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_m x_m)}{1 + exp\,(\alpha + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_m x_m)} \tag{2.4.1}$$

and

$$log\frac{\pi(x)}{1-\pi(x)} = \alpha + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_m x_m \tag{2.4.2}$$

where $\alpha, \beta = (\beta_1, \beta_2, \ldots, \beta_m)$ are constant parameters.

The transformation $L(x) = log\frac{\pi(x)}{1-\pi(x)}$ is referred to as the $log - odds$, or the $logit$, and is linearly related to x. The logit transformation links the problem to linear regression analysis. Methods based on maximum likelihood or weighted least-squares can be used to estimate the model parameters $\alpha, \beta_1, \beta_2, \ldots, \beta_m$ [50, 51]. The relative magnitudes of the parameters indicate the relative significances of the classifiers in their marginal contribution to the logit. Hence the parameters can be used as weights for the rank scores.

For each test pattern, the logit for each class is predicted by the estimated model. If only a ranking of the class set is needed, the classes can simply be sorted by the predicted logits in descending order. The class with the largest logit is then considered as most likely to be the true class. The values of $\pi(x)$ or the logit can also be used as a confidence measure. A threshold on these values can be determined experimentally, so that classes with confidences lower than the threshold can be rejected.

2.4.3 Support Vector Machine

Support Vector Machine (SVM) is originally designed for two-class classification problem, i.e., given a set of training samples, each marked as belonging to one of two categories, an SVM training algorithm builds a model that assigns new samples into one category or the other. The basic idea is to look for the maximum-margin hyperplane

which separates samples from category one and samples from category two as clearly as possible (as shown in Figure 2.9). The original optimal hyperplane algorithm proposed by Vapnik in 1963 [5] was a linear classifier. However, it is not always possible to find such a hyperplane. There are two approaches to solve this problem.

In 1992, Bernhard Boser, Isabelle Guyon and Vapnik suggested a way to create nonlinear classifiers by applying the kernel trick to maximum-margin hyperplanes [52]. The resulting algorithm is formally similar, except that every dot product is replaced by a nonlinear kernel function. This allows the algorithm to fit the maximum-margin hyperplane in a transformed feature space. The transformation may be nonlinear and the transformed space high dimensional; thus though the classifier is a hyperplane in the high-dimensional feature space, it may be nonlinear in the original input space.

Another approach is using soft margin. In 1995, Corinna Cortes and Vladimir Vapnik suggested a modified maximum margin idea that allows for mislabelled examples [53]. If there exists no hyperplane that can split the "yes" and "no" examples, the Soft Margin method will choose a hyperplane that splits the examples as cleanly as possible, while still maximizing the distance to the nearest cleanly split examples.

Although SVM is originally designed to be a classifier, it can also be used to combine multiple classifiers effectively in the case of two-class classification problem. Two-class classification problem can be stated as verifying whether two input samples belong to the same class or not. Assume that we have a number of classifiers, each producing a score representing distance (or similarity) between two input samples. These scores can be grouped into a vector which in turn is treated as a sample for SVM algorithm. The ensemble always performs better (or at least not worse) than a single classifier. This combination approach works best when the number of classifiers is not too large,

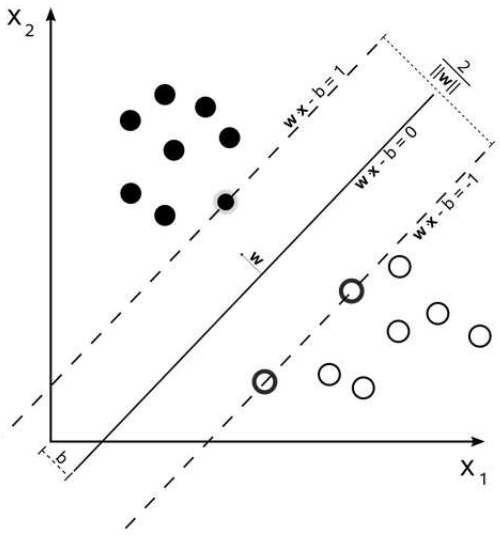

Figure 2.9: Maximum-margin hyperplane found by SVM algorithm (*taken from [5]*)

ideally under 30, because of the law of diminishing returns. In chapters 4, 6, 7, this approach will be used to combine multiple classifiers and achieve great results.

2.5 Performance Evaluation

There are two main problems in face recognition: face identification and face verification. Although they are very similar, there exist important differences requiring different techniques and datasets. In the next section, the distinction between face identification and face verification will be made clear.

2.5.1 Face Identification and Verification

A biometric system can be operated in two modes: identification mode and verification mode. In the identification mode, a biometric system establishes the identity of the user without a claimed identity while a biometric system operating in the verification

mode either accepts or rejects a user's claimed identity. Face identification is a more complicated problem than face verification because a large number of comparisons need to be performed for complete identification. An identification system needs to match the input face with all faces in the database, thus algorithm efficiency is a critical issue. On the other hand, a verification system needs to compare the input face with only one face corresponding to the claimed identity, and therefore has increased speed.

There are many potential applications for a biometric system working in identification mode or verification mode. One famous example of identification systems is visual surveillance in which people moving under public cameras need to be identified. For example of verification systems, an ATM system which verifies a user's face with a biometric upon each transaction would need to match only the current face image (acquired at point of transaction) with a single template stored on the ATM card. A typical face verification system can be divided into two modules: enrolment and verification. The enrolment module scans the face of a person through a sensing device and then stores a representation (template) of the face in the database. The verification module is invoked during the operation phase. The same representation used in enrolment phase is extracted from the input face and matched against the template of the claimed identity to provide a "yes/no" answer.

Because of the difference between identification and verification, they need to be evaluated in different ways.

2.5.2 Face Identification Evaluation

The performance of a face identification system is usually evaluated by its identification rate, which is calculated by matching a set of test face images with those in the database. Different algorithms can be evaluated by matching each test face image. The matching attempts performed for each test usually consist of correct matches and incorrect matches. A matching is considered as correct if the two face images being matched are from the same person, and incorrect otherwise. Identification rate is defined as the ratio between the number of correct matches and the number of test images.

2.5.3 Face Verification Evaluation

In a face verification system, system level performance evaluations are usually performed by cross matching the face images in the database. Different algorithms can be evaluated by matching each face image in the database with the rest of the images in the database. A threshold value is normally used in a way that a matching attempt is considered authentic when the matching score is equal to or above the threshold value. Two metrics (FAR and FRR) are used to measure performance of the whole system. The false acceptance rate, or FAR, is the measure of the likelihood that the biometric security system will incorrectly accept an access attempt by an unauthorised user. A system's FAR typically is stated as the ratio of the number of false acceptances divided by the number of impostor attempts. The false rejection rate, or FRR, is the measure of the likelihood that the biometric security system will incorrectly reject an access attempt by an authorised user. Analysis of the FAR shows how well the system can distinguish a correct match from an incorrect match and is usually related to the

uniqueness of the features. On the other hand, FRR analysis focuses on the repeatability of the features between different faces of the same person and is related to the reliability of the features.

A system can be tuned for a particular application by varying the value of these two metrics. A low value for both metrics is often desirable. Unfortunately, attempts to minimise FAR or FRR would compromise each of the metrics. For example, if the threshold is chosen high so that fewer impostors are falsely accepted by the system, FAR is getting lower. However, this also means that more authorised users are falsely rejected. In other words, FRR is getting higher, which is illustrated in Figure 2.10.

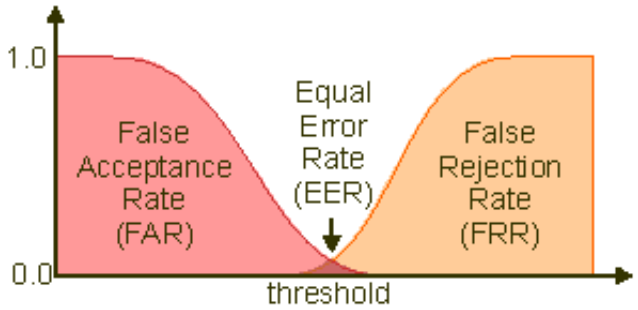

Figure 2.10: FAR, FRR curves and EER point

The Receiver Operating Curve (ROC) plots FAR versus FRR [54] for a system and can be used as a guide for the selection of an operating point for the system. In reporting the performance, the values of FAR and FRR for the ROC-curve are computed by varying the threshold value and using:

$$FAR = \frac{n_{ac}}{n_u}; \;\; FRR = \frac{n_{re}}{n_a} \tag{2.5.1}$$

In (2.5.1), n_a is the number of access attempt by an authorised user and n_u is the number of access attempt by an unauthorised user. For a given threshold value, n_{ac} is the number of acceptances and n_{re} is the number of rejections. From the ROC-curve, the Equal Error Rate (EER) is defined as the point where the value of FAR equals the value of FRR. The value of EER can now be used to determine the performance of the system. The lower the value of EER, the more reliable the system.

Now the difference between face identification and face verification evaluations has been discussed. Another problem in evaluation is how to make different algorithms' results directly comparable. Face datasets are created to address this issue.

2.6 Datasets

To provide a thorough evaluation of our proposed methods, we carried out experiments on four popular datasets: FERET, ORL, LFW, and MOBIO [55, 56, 57, 1, 58]. The first two, FERET and ORL, are designed for face identification and the last two, LFW and MOBIO, are for face verification. The aim of this section is to give an introduction to the history and design objectives of these datasets together with their protocols, i.e., how can they be used to evaluate face recognition algorithms.

2.6.1 FERET

Being created in 1993, FERET [55] was the first popular face dataset freely available to researchers. The FERET program ran from 1993 through 1997. Sponsored by the Department of Defense's Counterdrug Technology Development Program through the Defense Advanced Research Products Agency (DARPA), its primary mission was to

develop automatic face recognition capabilities that could be employed to assist security, intelligence and law enforcement personnel in the performance of their duties. The stated objective of the FERET program is to develop new techniques, technology, and algorithms for the automatic recognition of human faces. Research in face recognition has advanced considerably since then. Researchers have come close to fully recognizing all the frontal images in FERET [25, 26, 28, 59, 60]. Specifically, Shiguang et. al. reported in [25] that their system had achieved 99% recognition rates on the first two frontal subsets and about 90% on the last two.

Figure 2.11: The FERET sample face images

Originally, FERET contains only grayscale versions of images. Colour images were only released years later in 2003. The size of each image in the original form is 512×768

pixels. All images were taken from 1199 subjects. Figure 2.11 presents a number of face samples in the colour version. Although there are non-frontal face images in FERET, most published works focused on the frontal subset. According to the FERET protocol, frontal face images are split into smaller sets for training and testing purposes. They are a gallery set *fa* (1196); a training set (736); and four probe sets: *fb* (1195), *fc* (194), *Dup I* (722) and *Dup II* (234). Numbers inside the brackets are the number of face images in each set. Each probe set was mainly designed to test a specific aspect of an algorithm. For example, *fb* can be used to test the robustness to expression variations, *fc* for illumination, *Dup I* and *Dup II* for ageing. The *Dup I* probe images were obtained anywhere between one minute and 1031 days after their respective gallery matches. The harder *Dup II* probe images are a strict subset of the *Dup I* images; they are those taken only at least 18 months after their gallery entries. Among these sets, *Dup II* is the most difficult one in the FERET dataset.

2.6.2 ORL

Another popular face dataset designed to evaluate face identification algorithms is ORL. The ORL Database of Faces (ORL) created by AT&T Laboratories Cambridge, contains a set of face images taken between April 1992 and April 1994 in the lab. There are ten different images of each of 40 distinct subjects. So in total, there are 400 images from 40 subjects. For some subjects, the images were taken at different times, varying the lighting, facial expressions (open/closed eyes, smiling/not smiling) and facial details (glasses/no glasses). Both hair and forehead are included in the face images and the poses vary from left to right and up to down. In the original form, all images are grayscale and the size of each image is 92×112 pixels. Figure 2.12 presents a number

of samples in the ORL dataset.

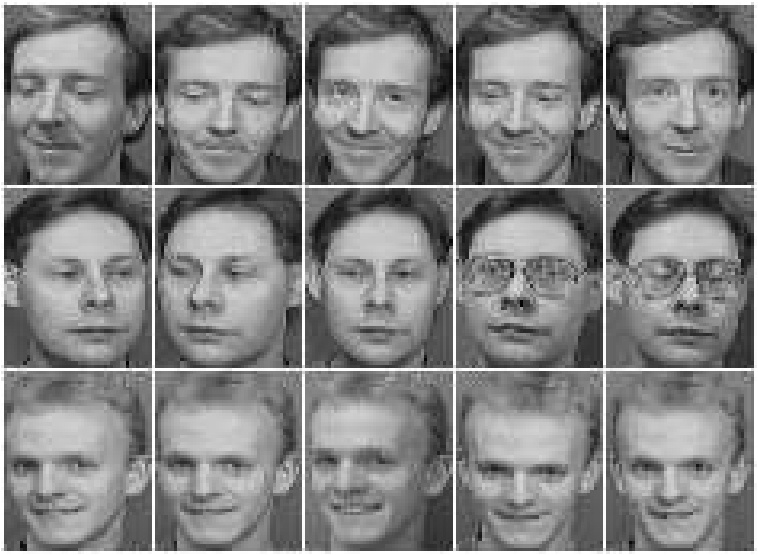

Figure 2.12: The ORL sample face images

There is no official protocol for ORL. The most popular way to evaluate face recognition algorithms on ORL is to randomly split images of each subject into two separate subsets: one for training and one for testing. For example, 3 images per subject for training and another 7 for testing can be used. In that case, there are a total of 120 (40×3) training images and 280 (40×7) testing images. In training phase, an algorithm can freely use 120 training images to build its learning model. Then in testing phase, 280 testing images are tested on that model to determine the recognition rate (i.e. accuracy) of the algorithm. Because of the randomness in the splitting process, the evaluation is often repeated multiple times to reduce numerical variation.

2.6.3 LFW

Recently a new face dataset called Labelled Faces in the Wild (LFW) [1] was created. LFW is a full protocol for evaluating face verification algorithms. Unlike FERET, LFW is designed for unconstrained face verification. Faces in LFW can vary in all possible ways due to pose, lighting, expression, age, scale, and misalignment. For that reason, LFW is currently considered to be the most challenging protocol in the field. The data set contains more than 13,000 images of faces collected from the web. Each face has been labeled manually with the name of the person pictured. 1680 of the people pictured have two or more distinct photos in the data set. Figure 2.13 presents a number samples in the LFW dataset. Because LFW will be used extensively in our experiments, its protocol is discussed in detail here.

Figure 2.13: The LFW sample face images

Data in LFW is organised into two views. View 1 is for algorithm development and general experimentation, prior to formal evaluation. This might also be called a model selection or validation view. The training set consists of 1100 pairs of matched images and 1100 pairs of mismatched images. The test set consists of 500 pairs of matched and 500 pairs of mismatched images. The people who appear in the training and testing sets are mutually exclusive. The main purpose of this view of the data is so that researchers can freely experiment with algorithms and parameter settings without worrying about overusing test data. For example, if one is using support vector machines and trying to decide upon which kernel to use, it would be appropriate to test various kernels (linear, polynomial, radial basis function, etc.) on View 1 of the database. To use this view, simply train an algorithm on the training set and test on the test set. This may be repeated as often as desired without significantly biasing final results.

View 2, for performance reporting, should be used only for the final evaluation of a method. The objective of this methodology is to use the final test sets as seldom as possible before reporting. Ideally, it should only be used once, as choosing the best performer from multiple algorithms, or multiple parameter settings, will bias results toward artificially high accuracy. The second view of the data consists of ten subsets of the database. Once a model or algorithm has been selected (using View 1 of the database if desired), the performance of that algorithm can be measured using View 2. To report accuracy results on View 2, the experimenter should report the aggregate performance of a classifier on 10 separate experiments in a leave-one-out cross validation scheme. In each experiment, nine of the subsets should be combined to form a training set, with the tenth subset used for testing. For example, the first experiment would use subsets (2, 3, 4, 5, 6, 7, 8, 9, 10) for training and subset 1 for testing. The fourth experi-

ment would use subsets (1, 2, 3, 5, 6, 7, 8, 9, 10) for training and subset 4 for testing. It is critical for accuracy performance reporting that the final parameters of the classifier under each experiment be set using only the training data for that experiment.

There are two ways to use training data: Image-Restricted Training or Unrestricted Training. The idea behind the Image-Restricted setting is that the experimenter should not use the name of a person to infer the equivalence or non-equivalence of two face images that are not explicitly given in the training set. Under the Image-Restricted training setting, the experimenter should discard the actual names associated with a pair of training images, and retain only the information about whether a pair of images is matched or mismatched. The idea behind the Unrestricted Training setting is that one may form as many pairs of matched and mismatched pairs as desired from a set of images labeled with individuals' names.

Most published methods operate on the Image-Restricted Training setting [8, 61, 62, 63, 64]. To make our results directly comparable with those of other methods, we also use the Image-Restricted Training setting in our experiments.

2.6.4 MOBIO

Another recently created database used to evaluate face verification algorithms is MOBIO. The mobile biometry (MOBIO) database was captured as part of the MOBIO project [58]. This project covers the use of two main forms of biometry for mobile authentication, these being: face and speech. To this end the MOBIO database, a multimodal face and speech database, was captured to reflect potential real-world scenarios for face and speech authentication on a mobile device. Figure 2.14 presents a number

of frames extracted randomly from videos in MOBIO. In the context of this thesis, we are only interested in the face verification aspect of this database.

Figure 2.14: The MOBIO sample extracted frames

In the ICPR 2010 Face Verification contest, only Phase I was used to evaluate participants' submitted algorithms. Phase I of the database consists of six sessions for 160 participants. Each session consists of 21 recordings. The database was captured at six separate sites in five different countries. These sites are at the University of Manchester (UMAN), University of Surrey (UNIS), Idiap Research Institute (IDIAP), Brno University of Technology (BUT), University of Avignon (LIA), and University of Oulu

(UOULU).

The database is being acquired primarily on a mobile phone. To address the concerns of both speech and face, researchers the participants were asked to answer a set of 21 questions which varied from (1) set responses, (2) read speech from a paper through to (3) free speech.

1. Set responses were given to the user. In total there were five such questions and fake responses were supplied to each user.

2. Read speech was acquired from each user by supplying the user with three sentences to read.

3. Free speech was acquired from each user by prompting the user with a random question. For five of these questions the user was asked to speak for five seconds and for ten questions the user was asked to speak for ten seconds, this gives a total of fifteen such questions. The user was again asked to not provide personal information and it was even suggested to not answer the question used to prompt them provided they could speak for the required time.

The database is split into three distinct sets: one for training, one for development, and one for testing. The splitting is such that two sites (in totality) are used for one split, which means there is no information about the individuals or the conditions for a site between sets. For the training set the data can be used in any way deemed appropriate and all of the data are available; normally the training set data would be used to derive background models (for instance, training a world background model UBM). The development set can be used to derive fusion parameters. However, it must be used to derive a threshold that is then applied to the test data. To facilitate this, the

development split and the test split both have the same style of protocol defined for them.

The protocol for the development split and the test split are the same. The first session is used to enrol the user but only the five set response questions can be used for enrolment. Testing is then conducted on each individual file for sessions two to six (there are five sessions used for development/testing) and only the free speech questions are used for testing. This leads to five enrolment videos for each user and 75 test client (positive sample) videos for each user (15 from each session). When producing impostor scores all the other clients are used, for instance if in total there were 50 clients then the other 49 clients would perform an impostor attack. For clarity the enrolment procedure and testing procedure are described again below.

- Enrolment data consists of the five set response recordings from the first session of the particular user.
- Testing data comes from the free speech recordings from every other session (the other five sessions) of the users, each video is treated as a separate test observation.

This dataset poses a very challenging problem. People in videos were not asked to look straight into the camera (as shown in Figure 2.14) therefore it is very hard to capture their full frontal faces. The low quality of videos captured by mobile devices makes the problem even harder. In chapter 7, our approach which was designed to work on MOBIO will be described.

Having reviewed the literature thoroughly, we start to present our contributions in the next chapter.

CHAPTER 3

Whitened Principal Component Analysis

This chapter addresses the first objective of the thesis: dealing with the lack of training data in face recognition. The worst case scenario is considered, when only a sample per person is available, and Whitened Principal Component Analysis (Whitened PCA) is proposed as a simple but effective solution. One major problem in face recognition is the difficulty of collecting training images. More samples usually mean better results but also more effort, time, and thus money. Unfortunately, many current face recognition techniques rely heavily on the large size and representativeness of training sets, and most methods suffer degraded performance or fail to work if there is only one training sample per person available.

3.1 Single Sample per Person Problem

Broadly speaking, single sample per person (SSP) problem is directly related to the small sample size problem in statistics and pattern recognition (see [6, 65] for a general discussion on this topic). As mentioned in the previous chapter, one important step of many face recognition methods is their subspace learning mechanisms. Unfortunately, the classic families of classifiers need sufficiently large training set for a good generalisation performance, partly due to the high-dimensional representation of face images. For example [66], for a 100×100 face image being vectorised into a 10000 dimensions feature space, theoretically the number of training images for each person should be at least ten times that of the dimensionality [6], that is, 100,00 images in total per person. It is practically impossible to collect such amount of training samples in many real life situations.

This SSP situation is common in face recognition. For example, consider an application in surveillance of public place such as airports and train stations where a large number of people need to be identified. One way to construct the needed face database efficiently is scanning photographs attached on most certificates such as passports, identification cards, student ID, driver license ID and so on, rather than really taking photos of each people. More often than not, one sample for each person in the database is provided. Therefore, a method which requires only one image per person in the database is desired.

On the other hand, storing only one sample per person in the database has several advantages, which are desired by most real world applications. Those are:

1. Easy to collect samples, either directly or indirectly: One common component

of face recognition systems is the face database, where the "template" face images are stored. Construction of such a face database is a very laborious and time-consuming work. This problem can now be effectively alleviated if only one image per person is required for sampling.

2. Save storage cost: The storage costs of face recognition system will reduce when only one image per person is needed in the database.

3. Save computational cost: The computational expense for large-scale applications could significantly reduce, because the number of training samples per person has direct effects on the costs of operations involved in face recognition, such as preprocessing, feature extraction, and recognition.

In summary, the above observations reveal that one sample problem is unavoidable in real world scenarios and it has equally significant advantages. To effectively solve the SSP problem, prior information needed to be taken into account. Therefore, it is very helpful to understand different types of variations in face images.

3.2 Variations in Face Images

In general, the difference between two faces can be modelled by three components: inherent difference that discriminates faces of different people; trained variation, arising from the different conditions of the same training face (class), such as expression and illumination changes; and novel variation, which is not characterised by the training samples, such as an unexpected accessory or illumination. Note that the variation of images of the same person consists of both the trained and the novel variation. In the

real-world scenario, since one never knows in advance the underlying distributions for the different faces, the novel variation is the most challenge factor. A good face recognition methodology should not only retain maximum inherent difference and minimum trained variation, but more importantly, also should be robust to the novel variation. Next PCA with whitening process (Whitened PCA) will be shown to be very good at both maximizing all the inherent difference in the training data (PCA step) and minimizing the trained variation (whitening step).

3.3 Whitened Principal Component Analysis

3.3.1 Standard PCA

A classical technique for dimensionality reduction, particularly in face recognition, is principle component analysis (PCA). In order to produce a compact representation, the feature vector is projected into a lower-dimensional feature space found by principle components analysis

$$\mathbf{u} = \mathbf{W_{pca}x} \tag{3.3.1}$$

The input vectors are first transformed by subtracting the mean: $\Phi_i = x_i - m$. The principal components of the training data set are given by the eigenvectors of its covariance matrix $\sum = \frac{1}{n}\sum_{i=1}^{n} \Phi_i \Phi_i^T$. In practice, only $M(M < n-1)$ eigenvectors having the largest eigenvalues (and, hence, the largest variance in the data set) are kept empirically to form the projection matrix W_{PCA}.

PCA technique is guaranteed to discover the linear projection that maximises the scatter of all the projected training samples, but this induces its main drawback for clas-

sification: the scatter being maximised is not only due to the inherent difference, but also due to the trained variation of the same class, which should be avoided. Note that in face recognition, the trained variations are usually subject to low frequency changes, such as the global variable lighting and similar expression changes of the training samples, which will be retained in leading components. Pentland et al. [67] have empirically shown that superior face recognition results are achieved when the first three eigenvectors are not used. It is unlikely that, however, the leading principal components corresponding solely to the trained variation; as a consequence, information that is useful for discrimination may be lost [68]. This problem is solved using the whitening process.

3.3.2 Whitening Process

PCA has two obvious shortcomings: (1) the leading eigenvectors encode mostly illumination and expression, rather than discriminating information [67]; and (2) Mean-Square-Error (MSE) principle underlying PCA favours low frequencies [69, 70, 71] and thus loses the discriminating information contained in the high frequency components. The whitening process normalizing the PCA based feature can directly counteract these disadvantages. Specifically, the PCA based feature, u is subjected to the whitening transformation and yields yet another feature set w:

$$\mathbf{w} = \mathbf{\Lambda_M^{-1/2}}\mathbf{u} \tag{3.3.2}$$

where $\Lambda_M^{-1/2} = diag\{\lambda_1^{-1/2}, \lambda_2^{-1/2}, \ldots, \lambda_M^{-1/2}\}$.

The integrated projection matrix $\Lambda_M^{-1/2} W_{PCA}$ treats variance along all principal component axes as equally significant by weighting components corresponding to smaller eigenvalues more heavily and is arguably appropriate for discrimination. Consequently, the negative influences of the leading eigenvectors are reduced while the discriminating details encoded in trailing eigenvectors are enhanced [27].

Whitened PCA is an improvement from PCA. Originally, PCA is not designed for classification. It is specially designed for dimension reduction. As a result, applying PCA to face recognition does not produce good results.

3.3.3 Distance Metric

Popular similarity measures include $L1$, $L2$, Mahalanobis distance, and Cosine Similarity measure. In the PCA-based feature space, it is proven that the Mahalanobis distance measure performs the best followed in order by $L1$, $L2$ distance and Cosine Similarity measure, because Mahalanobis distance counteracts the fact that simple distance measures, like $L1$ and $L2$ distance, in the PCA space weight preferentially for low frequencies [72, 71]. In Whitened PCA, however, this preference is equalised explicitly by the whitening process, which makes the Mahalanobis distance unnecessary. Instead, we should reconsider the optimal similarity measure according to its invariance with the image changes.

Note that when the novel variation, unseen in the training set, is projected onto the feature space, most energy of the training data will distribute over all the eigenvectors. This is because such variations are somewhat independent on the variance retained by the feature space. In other words, novel variation, projected into the feature space,

is inclined to evenly affect the projected scale on each component, and thus has more effect on the $L1$ and $L2$ distance rather than the vector angle [27]. Therefore, the Cosine Similarity measure, δ_{CSM} , which is invariant to change in scale, is employed to perform the nearest neighbour search in the feature space for face recognition.

$$\delta_{CSM}(w_1, w_2) = \frac{-w_1^T w_2}{\|w_1\|.\|w_2\|} \tag{3.3.3}$$

3.4 Experimental Results

In this section, the results of testing Whitened PCA on multiple face datasets and feature extraction methods will be presented. The objective is to examine the robustness and effectiveness of Whitened PCA under the SSP condition by comparing with the original PCA and LDA; and also examine the overfitting effect occurring in the LDA algorithm when there is a lack of training data.

3.4.1 Results on the FERET Dataset

Figure 3.1: The FERET sample normalised faces

In the first experiment, PCA, Whitened PCA, and LDA are tested on the FERET dataset with two feature extraction methods: Intensity values and Local Binary Patterns (chapter 2). All frontal faces in the FERET dataset are used. As a reminder, in the FERET protocol, there are a gallery set *fa* (1196); a training set (736); and four probe sets: *fb* (1195), *fc* (194), *dup I* (722) and *dup II* (234) (refer to Section 2.6 for more details). In the normalisation step, face images are all cropped and resized to 128×128 pixels, and aligned using the locations of the eyes given in the FERET dataset. Histogram equalisation is then applied to reduce the illumination effect (as shown in Figure 3.1). Finally, all faces are normalised to zero mean and unit variance. For PCA and Whitened PCA, only the gallery set is used as the training data. The training set consisting of 736 faces is used to train LDA.

Table 3.1 and Table 3.2 present recognition rates (%) of methods tested in the exper-

Method	fb	fc	dup I	dup II
PCA	75.7	15.9	34.9	16.2
Whitened PCA	83.2	73.7	50.3	37.2
LDA	76.8	18.0	35.3	16.5

Table 3.1: Testing Intensity values feature on the FERET dataset (%)

Method	fb	fc	dup I	dup II
PCA	84.4	30.9	47.1	26.1
Whitened PCA	95.0	72.0	68.0	56.0
LDA	82.1	55.8	50.0	32.6

Table 3.2: Testing Local Binary Patterns feature on the FERET dataset (%)

iment. As shown in both tables, Whitened PCA consistently outperforms PCA and LDA. LDA is better than PCA in most cases but the difference is not significant. In case of fc, although both PCA and LDA suffer from large variation in illumination, Whitened PCA is much more robust. Whitened PCA outperforms LDA in all cases even LDA has access to more information (from the training set). The reason is the problem of lacking training data which has been discussed. This problem causes the overfitting in the LDA algorithm. Also, LBP performs better than Intensity values in all cases. This is consistent with results published in the original paper about LBP in face recognition [21].

3.4.2 Results on the ORL Dataset

Similarly in the second experiment, PCA, Whitened PCA and LDA are tested on the ORL dataset with two feature extraction methods: Intensity values and Local Binary Patterns. Each image is resized to 128×128 pixels (as shown in Figure 3.2) and then normalised to zero mean and unit variance.

Figure 3.2: The ORL sample normalised faces

To evaluate the algorithms, we can split 10 images from each person randomly to two subsets: one for training, one for testing. There are 9 possible ways of splitting starting from 1 training-9 testing to 9 training-1 testing as shown in Table 3.3. Note that the results are not available at the bottom-left corner since LDA requires at least two samples per class.

As shown in Table 3.3 and Table 3.4, despite not explicitly taking advantage of training data labels, Whitened PCA still outperforms LDA consistently. Again, the reason is the problem of lacking training data leading to the overfitting effect in LDA. Even when there are 9 image samples per subject, it is still not enough for LDA to capture the wide range of variations in the ORL dataset. From these tables, it can be shown that LDA

Method	$1-9$	$2-8$	$3-7$	$4-6$	$5-5$	$6-4$	$7-3$	$8-2$	$9-1$
PCA	63.6	80.9	87.1	88.8	92.0	94.38	92.5	91.25	92.5
Whitened PCA	66.9	82.2	87.9	89.6	92.5	95.8	95.0	93.75	95.0
LDA	##	65.0	76.7	78.8	85.0	89.6	92.9	93.75	94.4

Table 3.3: Testing Intensity values feature on the ORL dataset (%)

does improve accuracy when the number of training samples per class increases. LDA starts to outperform PCA with about 6 or more training samples per class. It is likely that LDA will also outperform Whitened PCA given enough training samples. However, when there is limited training data, Whitened PCA is still the preferred choice. Unlike the case of the FERET dataset, in ORL the difference between LBP and Intensity values in terms of performance is insignificant.

Method	$1-9$	$2-8$	$3-7$	$4-6$	$5-5$	$6-4$	$7-3$	$8-2$	$9-1$
PCA	43.1	65.3	73.2	82.1	83.5	88.8	86.7	87.5	92.5
Whitened PCA	64.7	80.0	82.9	89.2	93.0	95.0	95.8	97.5	97.5
LDA	##	50.0	66.7	73.8	80.1	91.7	93.6	96.0	96.4

Table 3.4: Testing Local Binary Patterns feature on the ORL dataset (%)

3.5 Discussion and Conclusion

In order to illustrate the problem of insufficient training data, we solved the SSP problem. Through extensive experiments, we found Whitened PCA the best choice when either there is a lack of training data or the variations are too large to model. Whitened PCA on the FERET and ORL datasets is tested against the original PCA and LDA. The

results show that Whitened PCA outperforms both PCA and LDA consistently, even in cases where there are multiple labelled training samples that LDA can utilise.

CHAPTER 4

Compact Binary Patterns and Multi-patch Classifier

This chapter addresses the second objective of the thesis: solving accuracy-speed trade-off problem. Two methods to improve both accuracy and speed at the same time, or at least improve accuracy without compromising speed noticeably will be discussed. They are a new feature extraction method called Compact Binary Patterns and a novel method of combining multiple classifiers using Memetic Algorithm.

4.1 Motivation

Feature extraction methods are crucial to the success of face recognition systems. Popular mathematical transform based feature extraction methods include the Discrete Cosine Transform, the Wavelet Transform, etc. Among these, the Gabor Wavelets feature is one of the most successful. Many systems based on Gabor Wavelets have since been developed [73, 59, 74]. Unfortunately, methods using Gabor Wavelets suffer from its

highly redundant property, therefore very time-consuming to compute. In [28], an improved version of AdaBoost called MutualBoost is introduced to select a small number of highly discriminant Gabor features, leading to an impressive improvement in speed. It took MutualBoost only 4 seconds to recognise 200 face images in the FERET dataset [28]. However, since MutualBoost learns selective features from training data, it has the same problem of all learning algorithms, which is overfitting. A small number of Gabor features are also unlikely to capture all important information of the face. Therefore, a method which can quickly extract features from the entire face is still desirable.

Recently, the local binary patterns (LBP) operator has been proposed for face recognition [21]. The LBP is robust to illuminating variations because the features extracted are invariant to intensity variations. Moreover, because of its simplicity, the LBP is extremely fast [23]. However, under varying lighting conditions, the performance of LBP is unsatisfactory [48]. More recently, the LBP is enhanced by Gabor Wavelets to create the Local Gabor Binary Patterns (LGBP), which has achieved excellent recognition rates on the FERET database [24, 26]. The LGBP is however computationally expensive. Hence, the development of a fast and accurate face recognition system remains a challenge.

In this chapter, we introduce a novel facial feature extraction method, namely the Compact Binary Patterns (CBP), which is a generalisation and improvement of the LBP. CBP extracted features are then enhanced by the Whitened Principal Component Analysis (Whitened PCA). In recognition, a face image is divided into multiple patches and multiple classifiers are trained with a Memetic Algorithm to recognise the patches. In face verification, a SVM to combine the multiple classifiers is used. We show, with experimental results, that if the classifiers are ordered in descending order of their weights,

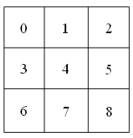

Figure 4.1: 3×3 block indexing

recognition rate can be improved significantly compared to the single classifier approach.

This chapter is structured as follows. The CBP feature is presented in Section 4.2, and the effectiveness of using Whitened PCA as a subspace learning method is discussed in Section 4.3. The Classifier Combination approach is described in Section 4.4 then experimental results are presented in Section 4.5. At the end, discussion and conclusion are given in Section 4.6.

4.2 Face Representation with Compact Binary Patterns

4.2.1 Local Binary Patterns

As discussed in chapter 2, the LBP operator [22] is a powerful operator for image texture description. The operator labels the pixels of an image by thresholding the 3×3-neighbourhood of each pixel against the value of the centre pixel and representing the result as a binary number. Histogram of the labels can then be used as a texture descriptor. An extension to the LBP operator is to the so called uniform patterns [3]. A LBP is called uniform if it contains at most two bitwise transitions from 0 to 1 or vice versa when the binary string is considered circular. Uniform patterns are often used for face recognition [23]. However, the LBP can be further improved.

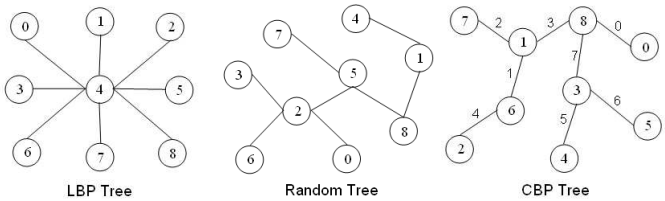

Figure 4.2: LBP, Random and CBP Tree

10	32	49
68	25	15
97	12	25

LBP: 01101011
CBP: 00110111

Figure 4.3: LBP and CBP examples

If we number each pixel in a 3×3 block as shown in Figure 4.1 and consider each pixel as a node in a 9-node tree, the LBP will be the concatenation of the 8 bits computed from the edges of the LBP tree in Figure 4.2, i.e. putting these bits together to create a 8-bit value. In other words, each edge on the tree corresponds to one bit in the 8-bit pattern. Each bit can be computed by comparing the values of two nodes forming the corresponding edge. From this angle, LBP corresponds to only one tree among the many possible trees. It is thus not clear if the LBP tree is the best one to represent the face images. From the Cayley's formula [75], the total number of trees is $9^7 = 4782969$ (generally n^{n-2} with n nodes). Our objective is to find the most compact tree that is also the most discriminant one.

We propose to improve the LBP by preserving the locality property and generalizing the uniform property of the LBP. The locality property is preserved by limiting all the binary comparison within 3×3 windows and the uniform property is generalised by the introduction of dominant patterns. In the feature extraction step, only the dominant

patterns will be used. However, we do not want to completely ignore non-dominant patterns. Instead, all of them are grouped into a single pattern. As a result, there are a total of 22+1 (for all non-dominant patterns) used patterns.

4.2.2 Compact Binary Patterns

Inspired by the notion of uniform patterns, we define compactness as the minimum number of distinct patterns which account for at least 90% of all patterns. This is similar to the definition of uniform patterns which account for 90% of all LBP patterns. These distinct patterns are called dominant patterns. For example, the tree in the middle of Figure 4.2 is a random tree (RDT) and in terms of compactness, the LBP tree is the same as the RDT tree, and the CBP tree is the most compact one found using an exhaustive search method. For each possible tree, its average compactness is computed using a large number of face images. In Table 4.1, the compactness of LBP, RDT and CBP are computed from all the faces in the FERET gallery. It can be seen from Figure 4.4 that edges in the CBP face are enhanced compared with those of the LBP and RDT faces.

Note that both the order of bits used to compute the local texture of each pixel and the direction each bit is computed (which node used for thresholding) do not matter. In fact, given any tree, if two pixels produce the same patterns (same bin in histogram), any change in the order of bits or the directions of thresholding does not invalidate the property that these two pixels produce the same patterns. As a result, changing order of bits (or directions) only changes the order of elements of final feature vectors which does not affect the final performance of the whole algorithm. As a result, any order of bits such as the order corresponding to numbers on edges of CBP Tree in Figure 4.2 can be used. Figure 4.3 illustrates how to compute Local Binary Pattern and Compact

Table 4.1: Average compactness of LBP, RDT and CBP estimated from 1196 images in FERET gallery

Pattern	Compactness
Local Binary Patterns	52.99
Random Tree Patterns	51.96
Compact Binary Patterns	21.96

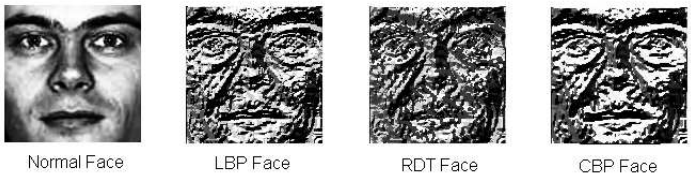

Figure 4.4: Normal Face, LBP Face, RTD Face and CBP Face

Binary Pattern of a pixel at the centre of 3×3 block.

4.2.3 Face Representation

The face image is divided into a grid then slide a 8×8 window over the edges of the grid as shown in Figure 4.5. From each 8×8 window, extract histograms of 23 CBP patterns can be extracted. Hence, the total number of features is $(\frac{H}{4} - 1) * (\frac{W}{4} - 1) * 23$ where H and W are height and width of the face image respectively. Our method is somewhat different from the method used in [23] where the grid unit is of a different size and the windows are non-overlap. After facial features have been extracted, we use the Whitened PCA for subspace learning.

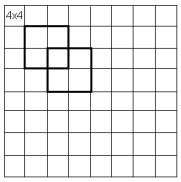

Figure 4.5: Face grid for extracting CBP features

4.3 Recognition with Subspace Learning

In the recognition step, the straightforward nearest-neighbour classifier can be used right after the feature vectors are extracted. Since feature vectors are essentially histogram, the dissimilarity (distance) between two feature vectors can be computed with either Histogram intersection, Log-likelihood statistic, or Chi square statistic (χ^2) as follows:

- Histogram intersection:

$$HI(A,B) = \sum_i min(A_i, B_i)$$

- Log-likelihood statistic:

$$LL(A,B) = -\sum_i A_i log B_i$$

- Chi square statistic (χ^2):

$$\chi(A,B) = \sum_i \frac{(A_i - B_i)^2}{A_i + B_i}$$

Among these metrics, Chi square statistic χ^2 was reported to perform best with LBP on the FERET dataset 21, 23. Although Chi square can also be used with CBP, it can be better by applying Whitened PCA as a subspace learning method before recognition.

There are two reasons why Whitened PCA should be preferable in this situation. First, since CBP is much more compact than LBP, it is likely that the discriminabilities of components of each feature vector are not uniformly distributed. While Chi square statistic treats components of the feature vector equally, Whitened PCA can detect more meaningful structure of data. Second, although we have a set of training samples (in both cases of the FERET and LFW datasets used in this chapter), it is better to defer using the training data to the next step where training data is used to learn the weights of multiple classifiers. Moreover, the amount of training data is limited. This is similar to the condition of lacking training data which has been discussed in chapter 3 in which Whitened PCA performs extremely well. In summary, Whitened PCA is proposed to use as a subspace learning method before recognition. As shown in section 4.5, Whitened PCA improves the accuracy of CBP considerably.

The next step is to make use of training data by learning a multi-patch classifier.

4.4 Classifier Combination Approach

4.4.1 Multiple Face Patches

The CBP+Whitened PCA method uses holistic representation of images, which has three disadvantages. First, in holistic representation the spatial information of facial features is not utilised. Second, facial features are used indiscriminately of their capacity for discrimination. Generally speaking, facial features such as eyes, nose, and mouth are considered to be more discriminative for face recognition. Thirdly, image variation due to pose and illumination changes within the whole image space is sometimes too large to be modelled by linear subspace methods such as PCA [76]. To resolve

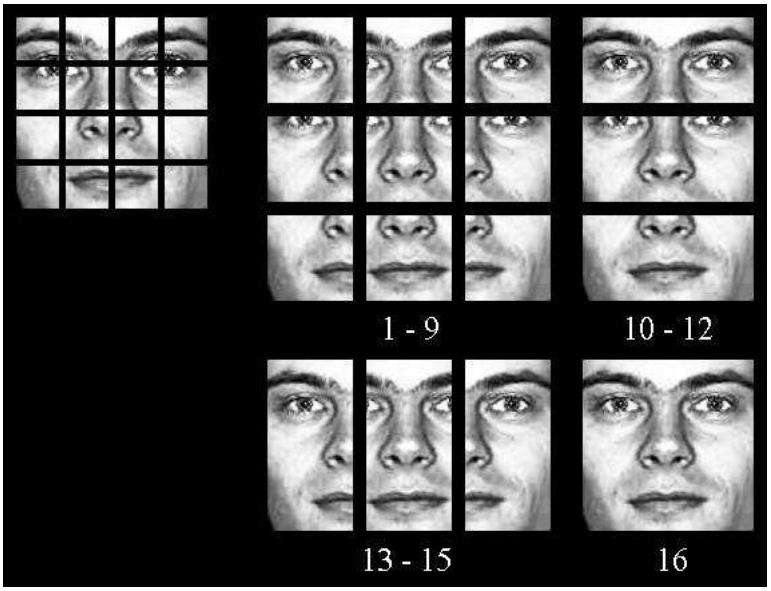

Figure 4.6: Multiple Patches from an example face

these problems, we propose a classifier combination system in which each classifier is responsible for a local region of the face.

The face image is divided into equal sized patches of varying resolution to obtain a total of 16 patches as shown in Figure 4.6 (though more advanced methods can be used to split the image). From each patch, a feature vector is extracted by CBP and Whitened PCA is used as the classifier. Consequently, we have a system of 16 classifiers. The next step is to combine these classifiers in an efficient way.

It is widely accepted that the classification accuracy can be improved by combining outputs of multiple classifiers [12]. How to combine multiple classifiers with various (potentially conflicting) decisions is still an open research topic. Since the contributions of different patches to classification results are different, it is sensible to have a weight value for each patch. While Boosting is a popular method to determine weights for multiple classifiers, it is more suitable in situations where there are a number of

weak classifiers which are slightly better than random guessing. Logistic Regression is another popular method to combine multiple classifiers [12].

For face verification, SVM is used to combine multiple classifiers. SVM is a standard technique; the reader is referred to [77] for more details. Given two input faces, each classifier produces a score as the similarity between the two faces. Then a p-dimension vector is formed by concatenating the scores from all p classifiers. This p-dimension vector is then passed to SVM for training and verification.

For face recognition, assuming we have p classifiers, the objective is to estimate p parameters $(\beta_0, \beta_1, ..., \beta_{p-1})$ in the linear form of the ensemble:

$$\beta_0 x_0 + \beta_1 x_1 + ... + \beta_{p-1} x_{p-1} \tag{4.4.1}$$

where β_i is the marginal contribution of the classifier i to the ensemble and x_i is the similarity score between a testing face and a face in gallery estimated by the classifier i. In this chapter, a Memetic Algorithm to estimate β_i is proposed.

4.4.2 Memetic Algorithm

The Memetic Algorithm (MA) [78] is a combination of Genetic Algorithm (GA) and Local Search. MA is proposed instead of GA because it has been known that in many optimisation problems, MA converges much quicker than GA. In algorithm 4.1, MA pseudocode used to train classifiers is presented.

Each chromosome is represented by an array of real numbers between 0 and 1 $(\beta_0, \beta_1, ..., \beta_{p-1})$. The i-th position of the array corresponds to the weight of the i-th classifier of the ensemble. The number of elements in the array is equal to the number of classifiers p. The fitness of a chromosome is defined as the recognition rate of the ensemble with the

Algorithm 4.1 Memetic Algorithm for training Multiple Patch Classifiers

```
Memetic_Algorithm() {
    initialise population P;
    repeat {
        for i = 1 to k {
            select two parents p1 and p2 from P;
            offspring_i = crossover(p1, p2);
            offspring_i = mutation(offspring_i);
            Hill_Climbing(offspring_i);
        }
        replace offspring_1,... and offspring_k in P;
    } until (stopping_condition);
    return the best solution;
}
```

weights represented by the chromosome. Note that by using the performance of the whole ensemble as fitness the diversity of the individual classifiers is also taken into account.

A population of 20 chromosomes is used. All positions of the chromosomes are set to random values between 0 and 1 at the beginning. To encourage the population diversity, the mutation operator is applied to all new chromosomes produced by the crossover operator. Normally, the mutation rate is set to be smaller than 10%. In practice, however, higher mutation rate sometimes can help the algorithm converge much more quickly and through experiments we found this to be true for our application. If the fitness values of the ten best chromosomes are similar the algorithm is terminated. Otherwise the algorithm is terminated after 100 generations. The weights of the chromosome with the highest fitness value during all generations (not only the last one) are the final result and are used for the weighted voting combination.

4.4.3 Hill Climbing

Genetic Algorithms are able to find global optimum by exploring a large search space when the selection pressure is properly controlled. However, the algorithms are weak with regard to fine-tuning near local optimum points, resulting in a long running time. That explains why Local Search is used in the Memetic Algorithms. Here, the popular Hill Climbing algorithm is used as the Local Search as shown in Algorithm 4.2.

Algorithm 4.2 Hill Climbing algorithm

```
Hill_Climbing(startSolution) {
    currentSolution = startSolution;
    loop {
        L = neighbours(currentSolution);
        nextFitness = - INF;
        nextSolution = NULL;
        for all x in L
            if Fitness(x)>nextFitness {
                nextSolution = x;
                nextFitness = Fitness(x);
            }
        if nextFitness<=Fitness(currentSolution) {
            return currentSolution;
        }
        currentSolution = nextSolution;
    }
}
neighbours(current) {
    resultSet = [];
    for i=1 to p {
        neighbour = current;
        change chromosome i of neighbour to a random number in [0,1];
        resultSet = resultSet + [neighbour];
    }
    return resultSet;
}
```

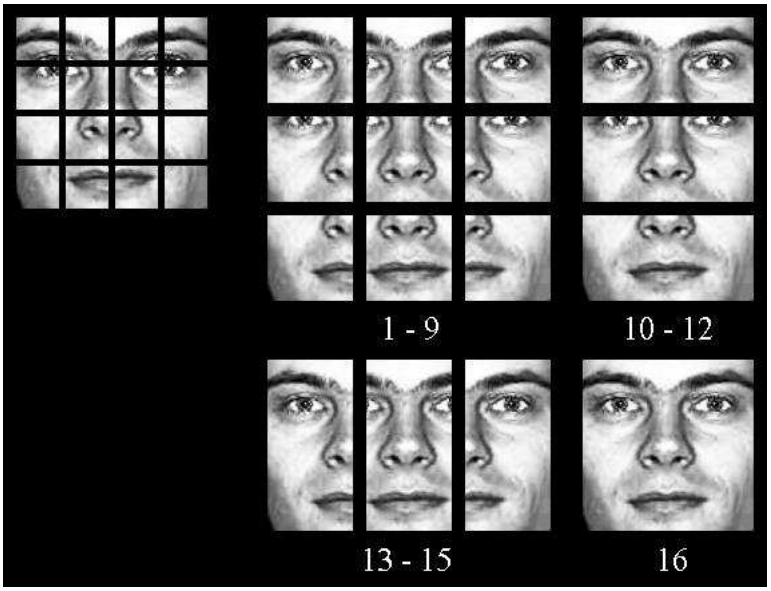

Figure 4.7: Multiple Patches from an example face

4.5 Experimental Results

In this section, the experimental results of our method on two popular datasets: the FERET and LFW datasets (refer to Section 2.6 for more details) will be presented. Although FERET was designed for face identification and LFW for face verification, we will show that our method can work well in both scenarios.

4.5.1 Results on the FERET dataset

In our experiments, we strictly evaluate all the methods using the standard gallery images (1196 images of 1196 subjects) and four probe sets, fb (1195 images), fc (194 images), dup I (722 images), and dup II (234 images). 736 frontal images of 314 subjects are used as the training set for methods that need a training stage. In the normalisation step, all face images are cropped and resized to 128×128 pixels, and aligned using

the locations of the eyes given in the FERET dataset. Histogram equalisation is then applied to reduce the illumination effect. Finally, all faces are normalised to zero mean and unit variance.

Probe\Patches	01	02	03	04	05	06	07	08
fb	92.2	94.1	93.5	66.7	76.8	66.3	51.7	54.5
fc	53.6	33.0	38.7	20.6	8.2	8.8	36.6	21.6
dup I	44.3	48.5	54.3	34.5	45.6	40.3	35.3	42.8
dup II	35.5	55.1	60.7	17.9	38.0	38.0	13.7	30.8

Probe\Patches	09	10	11	12	13	14	15	16
fb	50.9	**97.7**	78.8	63.6	88.0	93.2	88.5	95.6
fc	15.5	71.1	25.8	42.3	69.6	53.6	49.5	**80.4**
dup I	41.1	63.7	52.9	49.0	57.5	66.2	66.8	**73.7**
dup II	29.9	59.0	36.3	28.6	37.2	62.0	**66.2**	62.0

Table 4.2: Rank-1 recognition rates (%) of 16 patches on the FERET probe sets

Table 4.2 presents the recognition rates of 16 patches on all probe sets. Intuitively, larger patches perform better than smaller ones and patches corresponding with low variant areas (eyebrow for instance, patch 1, 2, 3, 10) perform better than high variant areas (mouth for instance, patch 7, 8, 9, 12). Numbers in Table 4.2 agree with the intuitiveness. Bold numbers indicate the best performance in each probe set. The largest patch (patch 16) do not perform best all the time. This suggests that sometimes it is better to ignore some noisy part of the face. However, when all these patches are combined, the performance is always improved (as shown in Table 4.3).

Methods	fb	fc	dup I	dup II
Fisherface	94.0	73.0	55.0	31.0
Best Results of [79]	96.0	82.0	59.0	52.0
Results of [21]	97.0	79.0	66.0	64.0
MutualBoost+GDA [28]	96.7	85.6	59.3	62.4
Weighted LGBPHS [24]	98.0	**97.0**	74.0	71.0
LBP+Whitened PCA	95.0	72.0	68.0	56.0
CBP+Whitened PCA	95.6	80.4	73.7	62.0
CBP+Whitened PCA+LR (16 patches)	97.7	84.0	76.3	73.5
CBP+Whitened PCA+MA (16 patches)	**98.4**	89.2	**80.6**	**76.5**

Table 4.3: Rank-1 recognition rates (%) of different algorithms on the FERET probe sets

To illustrate the effectiveness of the proposed Memetic Algorithm, we compare it with the Logistic Regression as described in [12]. In Table 4.3, our original method is called CBP+Whitened PCA+MA and the method using Logistic Regression is called CBP + Whitened PCA + LR. The experimental results show that CBP performs consistently better than LBP as a face descriptor and MA performs consistently better than LR as a Classifier Combination method. Best results are shown as bold numbers. Our method has achieved the best results for all datasets with only one exception: the fc set. In fact, the main performance difference between fb and fc is that in fc, the illumination varies a lot, and in some cases the variation is global which is too much for CBP (or LBP) to process. The best method for fc is the Weighted LGBPHS because Gabor filter

has proven to be robust to illumination variations [80]. Whitened PCA does not help much in this case because it maximises both types of variations: intrapersonal which is bad for recognition and extrapersonal which is good for recognition [26, 60]. In fc, intrapersonal (mainly illumination variation) is sometimes even larger than extrapersonal variation. In short, the difference in performance on the fc dataset between our method and Weighted LGBPHS is mainly because of the ability of the facial descriptors to cope with illumination variations. This may be a potential direction for further research.

4.5.2 Speed of the Multiple Patch Classifiers

We arrange the patches, and therefore the corresponding classifiers, in descending order of their weights. For first i patches, MA is used to estimate their weights which may be different from their weights in the ensemble of p classifiers. Figure 4.8 shows the curve with the number of patches in x-axis and the recognition rate in y-axis. As it can be seen, the recognition rate stabilises very quickly when the number of patches increases. Only 2 or 3 patches can be used to get very good result. In practice, a good strategy is to use as many patches as possible within the time limit. This is still true in fb, fc and dup I probe sets.

4.5.3 Results on the LFW dataset

The original size of each image is 250×250 pixels. At the preprocessing step, the image is simply cropped to remove the background, leaving a 80×150 face image. Three versions are available in LFW: original, funneled, and aligned. In [81], Wolf et al. showed

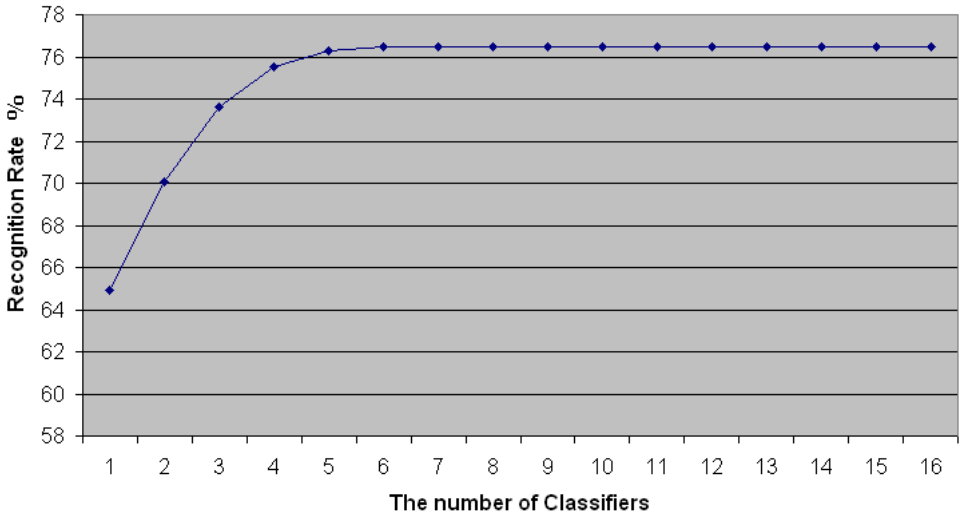

Figure 4.8: (Number of Patches, Recognition Rate) Curve for Dup II

that the aligned version is better than funneled version at dealing with misalignment. Hence, the aligned version will be used in all of our experiments. Figure 4.9 presents a number samples in the aligned version.

To form the covariance matrix for Whitened PCA step, we collected a set of faces from Caltech 10000 Web Faces database [82]. The dataset has 10,524 human faces of various resolutions and in different settings, e.g. portrait images, groups of people, etc. We do not use face images whose sizes are smaller than 30×30 pixels. Hence, the number of faces is reduced to 3,407. These 3,407 faces are normalised to the size of 80×150 (as shown in Figure 4.10). In our experiments, a thousand faces from these normalised faces are selected randomly to compute the covariance matrix.

CBP is used to extract features, Whitened PCA for subspace learning, and Cosine Similarity as distance measure. Table 4.4 presents the verification rates of 16 patches on the LFW dataset. Patch 16 (entire face) achieves the best performance of 0.78. However,

Figure 4.9: The LFW sample face images in the aligned version

Figure 4.10: Examples of normalised faces from Caltech 10000 Web Faces database

Patch	01	02	03	04	05	06	07	08
Rate	0.73	0.74	0.74	0.74	0.74	0.75	0.68	0.69
Patch	09	10	11	12	13	14	15	16
Rate	0.69	0.76	0.75	0.70	0.75	0.76	0.76	0.78

Table 4.4: Verification rates of 16 patches on the LFW dataset

Method	Verification Rate
LBP+OSS [81]	0.7820
Gabor+OSS [81]	0.7437
SIFT+LDML [38]	0.7927±0.0060
SIFT+ITML [38]	0.7620 ± 0.5
V1-like/MKL [83]	0.7935 ± 0.0055
CBP+Whitened PCA	0.7785 ± 0.0051
CBP+Whitened PCA+SVM	0.8162 ± 0.0051

Table 4.5: Verification rates of different algorithms on the LFW dataset

when all patches are combined together using SVM, the accuracy is improved to 0.8162 as presented in Table 4.5.

4.6 Discussion and Conclusion

In this chapter, CBP and Memetic Algorithm have been shown to be two very efficient methods in improving accuracy without compromising speed. The CBP has the same complexity as the LBP but has more discriminant power. In addition, some regions

of a face have less intrapersonal variations than others and for this reason a classifier combination approach for face recognition has been proposed. The weights of the classifiers are estimated by a Memetic Algorithm. Experiments show that CBP, Whitened PCA and MA work effectively together. The proposed method is not only accurate but also very fast because when patches are arranged in descending order of their weights, and only a few patches are needed to achieve the near optimal recognition result. The final classifier is tested thoroughly on the FERET and LFW databases. The results are comparable to the state of the art methods. In the next two chapters, we will focus on the third objective: the ability to recognise human faces within images reliably under unconstrained environments.

CHAPTER 5

Indirect Neighbourhood Components Analysis

This chapter and next chapter will introduce two novel metric learning methods which do not require any prior knowledge about data. In other words, they are designed to work under completely unconstrained conditions. Because the LFW dataset (Section 2.6) is the best one used to compare algorithms working under unconstrained conditions and also because it is designed for face verification (not face identification), we will apply our metric learning methods to face verification and test them on the LFW dataset. The method in this chapter, called Indirect Neighbourhood Components Analysis, is the combination of two recently introduced ideas: One-shot learning [8] and Neighbourhood Component Analysis [9]. Before discussing the method in detail, we briefly mention the motivation behind it.

5.1 Motivation

The biggest challenge in face verification comes from the numerous variations of face images, due to changes in lighting, pose, facial expression, and age. It is a very difficult problem, especially using images captured in totally uncontrolled environment, for instance, images from surveillance cameras, or from the Web. Over the years, many public face datasets have been created for researchers to advance state of the art and make their methods comparable. Recently, the LFW dataset has been published for studying the problem of unconstrained face recognition [1]. As a result, LFW is used in this chapter as a test bed for the problem of unconstrained conditions we are trying to solve.

Also, LFW is an open-set protocol in which subjects in testing phase are unknown in training phase [1] (refer to Section 2.6 for more details). This restriction prevents us from using the popular k-nearest neighbour (kNN) algorithm at the verification step. To overcome this problem, Wolf et al. proposed One-shot learning (OSL) method in which two testing faces are compared indirectly through faces in a negative set [8]. This method works very well and achieved state of the art results [81, 84]. The limitation of OSL is that it only uses training data for selecting the negative set and hence does not take full advantage of the training data.

Goldberger et al. [9] proposed Neighbourhood Components Analysis (NCA), a distance metric learning algorithm especially designed to improve kNN classification. The algorithm is to learn a Mahalanobis distance by minimizing the leave-one-out cross validation error of the kNN classifier on a training set. NCA was applied for face recognition and achieved very good performance [85]. However, NCA is not applicable to the

LFW dataset since it requires labelled training data (see Section 2.6).

Our proposed method tries to overcome the limitations of OSS and NCA by taking advantages of both. On the one hand, NCA is extended with an additional negative set of face images so that class labels are not required for training. On the other hand, a variant of the kNN algorithm particularly designed for verification is proposed. The method will be discussed in detail next.

5.2 Proposed Method

The main idea is that we want to apply NCA to face verification without the requirement of labelled training data. To achieve that objective, we propose to use an additional negative set of face images and a variant of kNN algorithm particularly designed for verification.

A negative set is a set of face images of subjects who do not appear in the training and testing sets. We denote this set Z (Figure 5.1) and its number of elements s. Then given two testing faces with feature vectors x and y, we can conclude they belong to the same person if x is closer to y than any z_j in Z and vice versa. In other words, x is y's nearest neighbour and y is x's nearest neighbour. This is similar to nearest-neighbour algorithm for classification.

To apply kNN to face verification, we have modified the classical kNN algorithm. First, all distances from y and z_j ($j = 1 \rightarrow s$) to x are sorted in ascending order and r_{xy} is defined as the ranking of the distance from y to x in that order. As there are $s + 1$ distances, r_{xy} can range from 1 to $s + 1$. Then x and y can be concluded to belong to the same person if $r_{xy} + r_{yx} \leq k$ ($k = 2$ in case of nearest-neighbour). With everything set

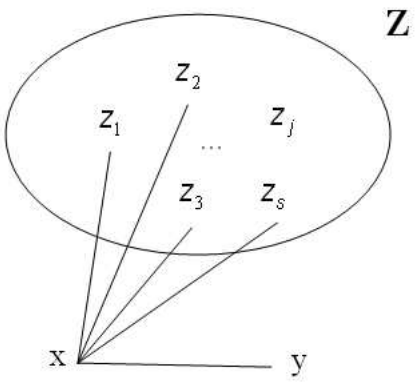

Figure 5.1: Negative set

up, we are ready to present the INCA method in detail.

We begin with the problem formalisation. Let $\{x_i, y_i, l_i\}$ denote a training set of samples with pairs of input vectors $x_i, y_i \in R^m$ and binary class labels $l_i \in \{1, 0\}$ which indicates whether x_i and y_i match or not. Note that class labels here are not the same as class labels NCA requires for training. In NCA, class labels provide information about subjects' identities. Then the objective of INCA is to learn a linear transformation $A : R^m \rightarrow R^d (d \leq m)$ which maximises the performance of kNN verification in the reduced subspace (refer to Section 2.3 for more information about metric learning).

Ideally, we would like to optimise performance on future test data, but since we do not know the true data distribution we attempt to minimise leave-one-out (LOO) performance on the training data instead. Given a finite set of linear transformations, we can easily select the best one, namely the one that minimises the number of verification errors. The kNN verification error, however, is a discontinuous function of the transformation matrix A, given that an infinitesimal change in A may change the neighbour graph and hence affect LOO verification performance by a finite amount. Therefore,

we cannot use this optimisation criterion in this case where there is a continuously parameterised family of linear transformations which must be searched. Instead, we adopt a more well-behaved measure of kNN performance, by using a differentiable cost function based on stochastic neighbour assignments in the transformed subspace. In particular, each input vector x_i selects y_i as its neighbour with some probability $p_{x_i y_i}$. Then we can compute the probability that x_i and y_i match as $\frac{1}{2}(p_{x_i y_i} + p_{y_i x_i})$. Similarly, the probability that x_i and y_i do not match can be computed as $1 - \frac{1}{2}(p_{x_i y_i} + p_{y_i x_i})$. We define $p_{x_i y_i}$ using a softmax over Euclidean distances in the transformed subspace:

$$p_{x_i y_i} = \frac{e^{-\|Ax_i - Ay_i\|^2}}{e^{-\|Ax_i - Ay_i\|^2} + \sum_{j=1}^{s} e^{-\left\|Ax_i - Az_j\right\|^2}} \tag{5.2.1}$$

Also, we denote the positive and negative sample index sets by Pos and Neg:

$$Pos = \{i | l_i = 1\}$$

$$Neg = \{i | l_i = 0\}$$

Under the stochastic selection rule (5.2.1), we can compute the probability p_i that sample i will be correctly verified:

$$p_i = \begin{cases} \frac{1}{2}(p_{x_i y_i} + p_{y_i x_i}) & if\ i \in Pos \\ 1 - \frac{1}{2}(p_{x_i y_i} + p_{y_i x_i}) & if\ i \in Neg \end{cases}$$

Therefore, the expected number of training samples correctly verified is:

$$\sum p_i = \sum_{i \in Pos} \frac{1}{2}(p_{x_i y_i} + p_{y_i x_i}) - \sum_{i \in Neg} \frac{1}{2}(p_{x_i y_i} + p_{y_i x_i}) + |Neg|$$

where $|Neg|$ is the number of negative samples. As $|Neg|$ and $\frac{1}{2}$ are constants, we can ignore them to have the following objective function:

$$f(A) = \sum_{i \in Pos} (p_{x_i y_i} + p_{y_i x_i}) - \sum_{i \in Neg} (p_{x_i y_i} + p_{y_i x_i}) \tag{5.2.2}$$

Differentiating $f(A)$ with respect to the transformation matrix A yields a gradient rule which we can use for learning. The gradient can be computed as follows:

$$\frac{\partial f}{\partial A} = \sum_{i \in Pos} \frac{\partial (p_{x_i y_i} + p_{y_i x_i})}{\partial A} - \sum_{i \in Neg} \frac{\partial (p_{x_i y_i} + p_{y_i x_i})}{\partial A}$$

$$\frac{\partial (p_{x_i y_i})}{\partial A} = \frac{1}{h_i(A)} \frac{\partial (g_i)}{\partial A} - \frac{g_i(A)}{h_i^2(A)} \frac{\partial (h_i)}{\partial A}$$

where $g_i(A)$ and $h_i(A)$ are the numerator and denominator of $p_{x_i y_i}$, that is:

$$\begin{aligned} g_i(A) =& e^{-\|Ax_i - Ay_i\|^2} \\ h_i(A) =& e^{-\|Ax_i - Ay_i\|^2} + \sum_{j=1}^{s} e^{-\left\|Ax_i - Az_j\right\|^2} \end{aligned}$$

We can continue with:

$$\begin{aligned} \frac{\partial (g_i)}{\partial A} =& - e^{-\|Ax_i - Ay_i\|^2} \times 2A(x_i - y_i)(x_i - y_i)^T \\ \frac{\partial (h_i)}{\partial A} =& \frac{\partial (g_i)}{\partial A} + \sum_{j=1}^{s} -e^{-\left\|Ax_i - Az_j\right\|^2} \times 2A(x_i - z_j)(x_i - z_j)^T \end{aligned}$$

As the roles of x_i and y_i are the same, $\frac{\partial (p_{y_i x_i})}{\partial A}$ can be computed similarly.

The learning algorithm is to maximise $f(A)$ using a gradient-based optimiser such as delta-bar-delta or conjugate gradients. We used the Conjugate Gradient method. Of course, as the objective function $f(A)$ is not convex, some care must be taken to avoid local maxima during training. We have experimentally observed that the linear transformation obtained by Principal Component Analysis (PCA) method can serve as a good starting point for the Conjugate Gradient algorithm.

Note that the norm of matrix A controls the softness of the neighbour assignments. By learning the overall scale of A as well as the relative directions of its rows we are also effectively learning a real-valued estimate of the optimal number of neighbours (k). For example, replacing A with αA, it can easily be shown that as α tends to infinity, the probabilistic assignment is reduced to deterministic nearest-neighbour assignment in the same transformed subspace. In practice, however, it is simpler to estimate the optimal value of k using cross validation. Algorithm 5.1 describes the proposed method with both training and testing phases. How INCA can be applied to face verification will be presented next.

Algorithm 5.1 The INCA method for verification

Training:

INPUT

- $S = \{x_i, y_i, l_i\}$: a set of training samples ($x_i, y_i \in R^m$, $l_i \in \{0, 1\}$)
- $Z = \{z_j\}$: a negative set of face images
- d: reduced dimension

OUTPUT

- $A_{d \times m}$: output transformation matrix that maximises the objective function (5.2.2)
- k: optimal value of k

1. set initial value for A (e.g. using the PCA method)
2. apply the Conjugate Gradient method to maximise the objective function
3. estimate k using cross validation
4. **return** A and k

Testing:

INPUT - (x, y): two testing faces

OUTPUT - match/unmatch decision

1. transform x and y to dimension-reduced subspace using the INCA transformation matrix
2. verify using the proposed kNN variant

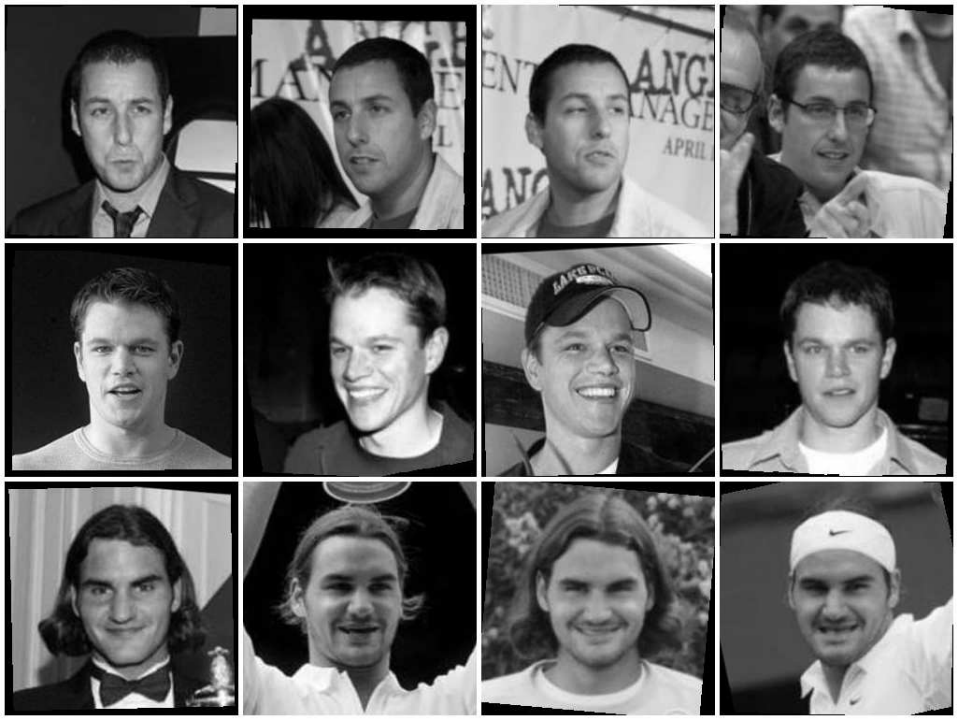

Figure 5.2: The LFW sample face images in the aligned version

5.3 Application to Face Verification

In this section, we show how INCA can be applied to face verification on the LFW dataset in detail.

5.3.1 The LFW dataset

Three versions are available in LFW: original, funneled, and aligned. In [81], Wolf et al. showed that the aligned version is better than funneled version at dealing with misalignment. Hence, the aligned version will be used in all of our experiments. Figure 4.9 presents a number sample images in the aligned version.

5.3.2 The negative set

The negative set is collected from Caltech 10000 Web Faces database [82]. The dataset has 10,524 human faces of various resolutions and in different settings, e.g. portrait images, groups of people, etc. We do not use face images whose sizes are smaller than 30×30 pixels. Hence, the number of faces is reduced to 3,407. These 3,407 faces are normalised to the size of 80×150 (as shown in Figure 5.3). In our experiments, a thousand faces from these normalised faces are selected randomly to form the negative set.

Figure 5.3: Examples of normalised faces from Caltech 10000 Web Faces database

$13,000$

5.3.3 Face verification pipeline

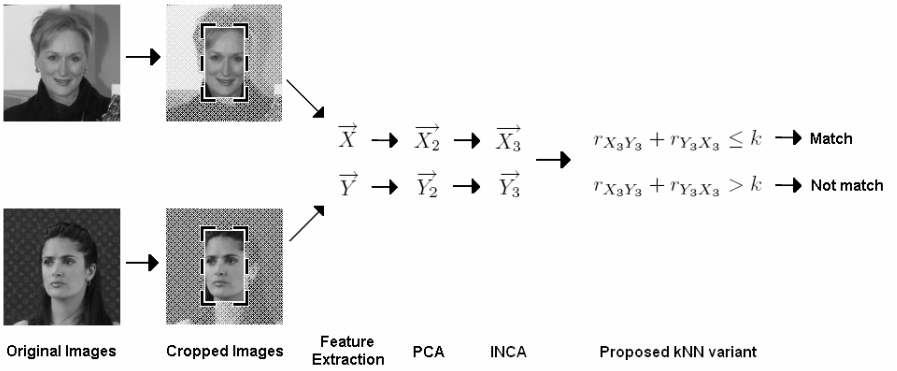

Figure 5.4: The INCA face verification pipeline

The overview of our method is presented in Figure 5.4. First, two original images are cropped to smaller sizes. Next some feature extraction method is used to form feature vectors $(\overrightarrow{X}, \overrightarrow{Y})$ from the cropped images. These vectors are passed to PCA to get two dimension-reduced vectors $(\overrightarrow{X_2}, \overrightarrow{Y_2})$. Then INCA is used to transform $(\overrightarrow{X_2}, \overrightarrow{Y_2})$ to $(\overrightarrow{X_3}, \overrightarrow{Y_3})$ in the final subspace. Finally, the proposed kNN variant is used to conclude whether the two faces match or not. Each step will be discussed in detail.

5.3.3.1 Preprocessing

The original size of each image is 250×250 pixels. At the preprocessing step, the image is simply cropped to remove the background, leaving a 80×150 face image. The next step after preprocessing is to extract features from the image.

5.3.3.2 Feature Extraction

To test the robustness of our method to different types of features, we carry out experiments on three facial descriptors: Intensity, Local Binary Patterns and Gabor Wavelets.

Intensity is the simplest feature extraction method. The feature vector is formed by concatenating all the pixels. The length of the feature vector is $12,000$ ($= 80 \times 150$).

Local Binary Patterns (LBP) was first applied for face recognition in [21] with very promising results. In our experiments, the face is divided into non-overlapping 10×10 blocks and LBP histograms are extracted in all blocks to form the feature vector whose length is $7,080$ ($= 8 \times 15 \times 59$).

Gabor Wavelets [18, 80] with 5 scales and 8 orientations are convoluted at different pixels selected uniformly with the downsampling rate of 10×10. The length of the feature vector is $4,800$ ($= 5 \times 8 \times 8 \times 15$) .

5.3.3.3 Dimension Reduction

Before applying any learning method, we use PCA to reduce the dimension of the original feature vector to a more tractable number. A thousand normalised faces from the Caltech 10000 Web Faces database are randomly selected to create the covariance matrix in PCA. We notice in our experiments that the specific value of the reduced dimension after applying PCA does not affect the accuracy very much as long as it is not too small.

		Euclidean	PCA	OSS-LDA	INCA
IN	original	0.655 ± 0.0067	0.6587 ± 0.007	0.6867 ± 0.0059	0.7423 ± 0.0032
	sqrt	0.6535 ± 0.0061	0.6593 ± 0.0062	0.6718 ± 0.0057	0.7576 ± 0.0057
LBP	original	0.6527 ± 0.0098	0.6767 ± 0.0071	0.7335 ± 0.0051	0.8005 ± 0.0054
	sqrt	0.6977 ± 0.0047	0.7005 ± 0.0062	0.7617 ± 0.0035	$\mathbf{0.8217 \pm 0.0046}$
GABOR	original	0.5887 ± 0.0095	0.6083 ± 0.0103	0.7011 ± 0.0057	0.7848 ± 0.0035
	sqrt	0.6335 ± 0.0097	0.6552 ± 0.0075	0.7225 ± 0.0042	0.7916 ± 0.0049

Table 5.1: Mean ($\pm$ standard error) scores on the LFW using different methods

5.4 Experimental Results

In this section, the results of two experiments will be presented.

The objective of the first experiment is to test the performance of four methods on three types of features. Four tested methods are Euclidean distance in the original space (Euclidean), Principal Component Analysis (PCA), One-shot Similarity Learning using LDA (OSS-LDA) [8] and our method (INCA). Three types of features are Intensity (IN), Local Binary Patterns (LBP), and Gabor Wavelets (GABOR). As shown in Table 5.1, INCA improves about $5-8\%$ over OSS-LDA and about $10-15\%$ over Euclidean distance. LBP seems to perform better than Intensity and Gabor Wavelets. Using square root of the feature vector improves the accuracy about $2-3\%$ in most cases. The highest accuracy which can be achieved from a single type of feature is 0.8217 ± 0.0046 using INCA with the square root of the LBP feature.

The objective of the second experiment is to test how our method performs in different reduced dimensions. In this experiment, PCA and INCA are tested with the square

root of the LBP feature. The reduced dimensions range from 5 to 100. As shown in Figure 5.5, our method performs well even in 5-dimensional subspace (about 0.72, i.e. 72%) compared to PCA (about 0.60, i.e. 60%).

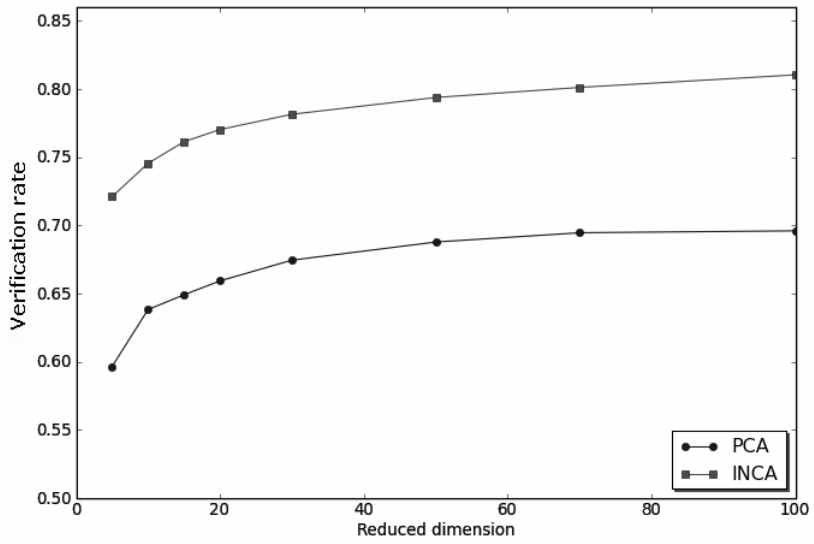

Figure 5.5: Mean scores on the LFW using PCA and INCA with different reduced dimensions

5.5 Discussion and Conclusion

A novel method called the Indirect Neighbourhood Components Analysis has been introduced for learning a distance metric based on the ideas of One-shot learning and Neighbourhood Components Analysis. Our method uses an addition set of faces images to remove the requirement of labelled training data in NCA and a kNN variant for verification. Our method is tested on the LFW dataset and achieved good results, even in very low-dimensional representations. Next our second metric learning method designed to solve the problem of unconstrained face recognition will be presented.

CHAPTER 6

Cosine Similarity Metric Learning

In the last chapter, a new metric learning method called Indirect Neighbourhood Component Analysis (INCA) is introduced. INCA is designed to solve the problem of unconstrained face verification and achieved very promising results. However, there are two aspects of INCA which we can improve further. First, it takes a long time to perform the training step in INCA since the objective function is highly complex. Second, the verification rate is good but still below state-of-the-art methods [10]. In this chapter, another novel metric learning method called Cosine Similarity Metric Learning (CSML) is introduced. Like INCA, CSML is designed for unconstrained face verification. CSML makes use of Cosine Similarity instead of the more popular Euclidean Distance. As a result, the objective function is simple thus very fast to optimise. Experimental results show that CSML produces the best results on the LFW dataset (Section 2.6) in the literature. Let us discuss the practical motivation behind CSML first.

Figure 6.1: From FERET to LFW

6.1 Motivation

As mentioned in Section 2.6, FERET [55] is the first popular face dataset freely available to researchers. Researchers have come very close to fully recognizing all the frontal images in FERET [25, 26, 28, 59, 60]. Although near perfect results have been achieved on the FERET dataset, these methods are not robust enough to deal with non-frontal face images.

Unlike FERET, LFW is designed for unconstrained face verification. As shown in Figure 6.1, faces in LFW can vary in all possible ways due to pose, lighting, expression, age, scale, and misalignment. Methods for frontal images cannot cope with these variations and as such many researchers have turned to machine learning to develop learning based face verification methods [38, 84]. One of these approaches is to learn a transformation matrix from the data so that the Euclidean distance can perform better in the new subspace. Learning such a transformation matrix is equivalent to learning a Mahalanobis metric in the original space [9] (refer to Section 2.3 for a brief proof). In chapter 2, three popular metric learning methods: Neighbourhood Component Analy-

sis, Large Margin Nearest Neighbour, and Information-Theoretic Metric Learning have been reviewed.

Our Cosine Similarity Metric Learning method is different from all the above methods in terms of distance measures. All of the other methods use Euclidean distance to measure the dissimilarities between samples in the transformed space whilst our method uses Cosine Similarity. Now let us present CSML in detail.

6.2 Cosine Similarity Metric Learning

The general idea is to learn a transformation matrix from training data so that Cosine Similarity performs well in the transformed subspace. The performance is measured by cross validation error (cve).

6.2.1 Cosine similarity

Cosine similarity (CS) between two vectors x and y is defined as:

$$CS(x, y) = \frac{x^T y}{\|x\| \, \|y\|}$$

Cosine similarity has a special property that makes it suitable for metric learning: the resulting similarity measure is always within the range of -1 and $+1$. As shown in section 6.2.3, this property allows the objective function to be simple and effective.

6.2.2 Metric learning formulation

Let $\{x_i, y_i, l_i\}_{i=1}^{s}$ denote a training set of s labelled samples with pairs of input vectors $x_i, y_i \in R^m$ and binary class labels $l_i \in \{1, 0\}$ which indicates whether x_i and y_i match

or not. The objective is to learn a linear transformation $A : R^m \rightarrow R^d (d \leq m)$, which we will use to compute cosine similarities in the transformed subspace as:

$$CS(x, y, A) = \frac{(Ax)^T (Ay)}{\|Ax\| \, \|Ay\|} = \frac{x^T A^T A y}{\sqrt{x^T A^T A x} \sqrt{y^T A^T A y}}$$

Specifically, we want to learn the linear transformation that minimises the cross validation error when similarities are measured in this way. We begin by defining the objective function.

6.2.3 Objective function

First, we define positive and negative sample index sets Pos and Neg as:

$$Pos = \{i | l_i = 1\}$$

$$Neg = \{i | l_i = 0\}$$

Also, let $|Pos|$ and $|Neg|$ denote the numbers of positive and negative samples. We have $|Pos| + |Neg| = s$ - the total number of samples.

Now the objective function $f(A)$ can be defined as:

$$f(A) = \sum_{i \in Pos} CS(x_i, y_i, A) - \alpha \sum_{i \in Neg} CS(x_i, y_i, A) - \beta \|A - A_0\|^2$$

We want to maximise $f(A)$ with regard to matrix A given two parameters α and β where $\alpha, \beta \geq 0$. The objective function can be split into two terms: $g(A)$ and $h(A)$ where

$$g(A) = \sum_{i \in Pos} CS(x_i, y_{i,} A) - \alpha \sum_{i \in Neg} CS(x_i, y_i, A)$$

$$h(A) = \beta \left\| A - A_0 \right\|^2$$

The role of $g(A)$ is to encourage the margin between positive and negative samples to be large. A large margin can help to reduce the training error. $g(A)$ can be seen as a simple voting scheme from each sample. The reason we can treat votes from samples equally is that Cosine Similarity function is bounded by 1. Additionally, because of this simple form of $g(A)$, we can optimise $f(A)$ very fast (details in section 6.2.5). The parameter α in $g(A)$ is to balance out the contributions of positive samples and negatives samples to the margin. In practice, α can be estimated using cross validation or simply be set to $\frac{|Pos|}{|Neg|}$. In the case of LFW, because the numbers of positive and negative samples are equal, we simply set $\alpha = 1$.

The role of $h(A)$ is to regularise matrix A to be as close as possible to a predefined matrix A_0 which can be any matrix. The idea is both to inherit good properties from matrix A_0 and to reduce the training error as much as possible. If A_0 is carefully chosen, the learned matrix A can achieve small training error and good generalisation ability at the same time. The parameter β plays an important role here. It controls the tradeoff between maximizing the margin ($g(A)$) and minimizing the distance from A to A_0 ($h(A)$).

With the objective function set up, the algorithm can be presented in detail in the next section.

6.2.4 The algorithm

The idea is to use cross validation to estimate the optimal values of (α, β). In this chapter, α can be simply set to 1 and suitable β can be found using coarse-to-fine search strategy. Coarse-to-fine means the range of searching area decreases over time. Algorithm 6.1 presents the proposed CSML method. Next we will prove that in theory the performance of learned matrix A_{CSML} is never worse than that of matrix A_0. In practice, however, the performance of matrix A_{CSML} is significantly better in most cases (see section 6.4).

Let's define $cve(A)$ as the function which receives matrix A and returns the cross validation error on validation samples when using A as the transformation matrix . Let's define two other functions op, cve_op as:

$$op(A_0, \alpha, \beta) = arg\max_{A} f(A)$$

$$cve\_op(\alpha, \beta) = cve(op(A_0, \alpha, \beta))$$

In words, op is a function which receives 3 input parameters (A_0, α, β) and returns the optimal matrix A^* maximizing $f(A)$ ($A^* = op(A_0, \alpha, \beta)$). cve_op is a function which receives 2 input parameters (α, β) and returns the cross validation error using the matrix returned by $op(A_0, \alpha, \beta)$.

Let's define (α^*, β^*) and A_{op} as:

$$(\alpha^*, \beta^*) = arg\,min_{\alpha,\beta}\ cve\_op(\alpha, \beta)$$

Algorithm 6.1 Cosine Similarity Metric Learning

INPUT

- $S = \{x_i, y_i, l_i\}_{i=1}^{s}$: a set of training samples ($x_i, y_i \in R^m$, $l_i \in \{0, 1\}$)
- $T = \{x_i, y_i, l_i\}_{i=1}^{t}$: a set of validation samples ($x_i, y_i \in R^m$, $l_i \in \{0, 1\}$)
- d: dimension of the transformed subspace ($d \leq m$)
- A_p: a predefined matrix ($A_p \in R^{d \times m}$)
- K: K-fold cross validation

OUTPUT - A_{CSML}: output transformation matrix ($A_{CSML} \in R^{d \times m}$)

1. $A_0 \leftarrow A_p$
2. $\alpha \leftarrow \frac{|Pos|}{|Neg|}$
3. **Repeat**
   (a) $min\_cve \leftarrow \infty$ // store minimum cross validation error
   (b) **For** each value of β // coarse-to-fine strategy
      i. $A^* \leftarrow$ the matrix maximizing $f(A)$ given (A_0, α, β) evaluating on S
      ii. **if** $cve(T, A^*, K) < min\_cve$ **then** // Algorithm 6.2
         A. $min\_cve \leftarrow cve(T, A^*, K)$
         B. $A_{next} \leftarrow A^*$
   (c) $A_0 \leftarrow A_{next}$
4. **Until** convergence
5. $A_{CSML} \leftarrow A_0$
6. **Return** A_{CSML}

Algorithm 6.2 Cross validation error computation

INPUT

- $T = \{x_i, y_i, l_i\}_{i=1}^{t}$: a set of validation samples ($x_i, y_i \in R^m$, $l_i \in \{0, 1\}$)
- A: a linear transformation matrix ($A \in R^{d \times m}$)
- K: K-fold cross validation

OUTPUT - cross validation error

1. Transform all samples in T using matrix A
2. Partition T into K equal-sized subsamples
3. $total\_error \leftarrow 0$
4. **For** $k = 1 \rightarrow K$

   // using subsample k as testing data, the other $K - 1$ subsamples as training data

   (a) $\theta \leftarrow$ the optimal threshold on training data

   (b) $test\_error \leftarrow$ error on testing data

   (c) $total\_error \leftarrow total\_error + test\_error$

5. **Return** $total\_error / K$

$$A_{op} = op(A_0, \alpha^*, \beta^*)$$

In words, (α^*, β^*) are parameters achieving minimum cross validation error. A_{op} is the matrix maximizing $f(A)$ when $\alpha = \alpha^*$ and $\beta = \beta^*$. Now we are ready to prove a simple but powerful theorem.

For every A_0 there always exists some (α_0, β_0) so that $cve\_op(\alpha_0, \beta_0) = cve(A_0)$.

Simply set $\alpha_0 = 1$, then function $g(A)$ becomes

$$\sum_{i \in Pos} CS(x_i, y_i, A) - \sum_{i \in Neg} CS(x_i, y_i, A)$$

Because $-1 \leq CS(x_i, y_i, A) \leq 1$, we have

$$-s = -|Pos| - |Neg| \leq g(A) \leq |Pos| + |Neg| = s$$

In other words, $g(A)$ is bounded by s. Therefore, we have

$$f(A) = g(A) - \beta \left\| A - A_0 \right\|^2 \leq s - \beta \left\| A - A_0 \right\|^2$$

$\longrightarrow$ { set $A \leftarrow A^*$ }

$$f(A^*) \leq s - \beta \left\| A^* - A_0 \right\|^2$$

$\longrightarrow$

$$\lim_{\beta \to \infty} \left\| A^* - A_0 \right\|^2 = 0 \tag{6.2.1}$$

because otherwise the term $\beta \|A^* - A_0\|^2$ can become indefinitely large which causes $f(A^*)$ to be indefinitely small. This is not true because we always have

$$f(A^*) \geq f(A_0) = g(A_0) \geq -s$$

From Eq (6.2.1), we have

$$\lim_{\beta \to \infty} A^* = A_0$$

$\longrightarrow$ { $A^* = op(A_0, \alpha_0, \beta)$ }

$$\lim_{\beta \to \infty} op(A_0, \alpha_0, \beta) = A_0$$

$\longrightarrow$ { definition of cve_{op} function and set β_0 to a very big number}

$$\exists \beta_0 \; so \; that \; cve\_op(\alpha_0, \beta_0) = cve(A_0)$$

$cve(A_0) \geq cve(A_{op}) \; \forall A_0$. As a result, we have

$$
\begin{aligned}
& cve(A_0) \\
=&\{from\ the\ above\ theorem\} \\
& cve\_op(\alpha_0, \beta_0) \\
=&\{definition\ of\ cve\_op\} \\
& cve(op(A_0, \alpha_0, \beta_0)) \\
\geq&\{definition\ of\ (\alpha^*, \beta^*)\} \\
& cve(op(A_0, \alpha^*, \beta^*)) \\
=&\{definition\ of\ A_{op}\} \\
& cve(A_{op})
\end{aligned}
$$

This means that the performance of the optimal matrix A_{op} is never worse than that of matrix A_0. Therefore, we can apply the process repeatedly (i.e. set $A_0 \leftarrow A_{op}$ after each iteration) until there is no further improvement (as shown in Algorithm 6.1).

6.2.5 Complexity Analysis

$f(A)$ is differentiable with regard to matrix A so we can optimise it using a gradient based optimiser such as delta-bar-delta or conjugate gradients. We used the Conjugate Gradient method. The gradient of $f(A)$ can be computed as follows:

$$\frac{\partial f(A)}{\partial A} = \sum_{i \in Pos} \frac{\partial CS(x_i, y_i, A)}{\partial A} - \alpha \sum_{i \in Neg} \frac{\partial CS(x_i, y_i, A)}{\partial A} - 2\beta(A - A_0) \tag{6.2.2}$$

$$\begin{aligned} \frac{\partial CS(x_i, y_i, A)}{\partial A} &= \frac{\partial(\frac{x_i^T A^T A y_i}{\sqrt{x_i^T A^T A x_i}\sqrt{y_i^T A^T A y_i}})}{\partial A} \\ &= \frac{\partial(\frac{u(A)}{v(A)})}{\partial A} \\ &= \frac{1}{v(A)}\frac{\partial u(A)}{\partial A} - \frac{u(A)}{v(A)^2}\frac{\partial v(A)}{\partial A} \end{aligned} \tag{6.2.3}$$

where

$$\frac{\partial u(A)}{\partial A} = A(x_i y_i^T + y_i x_i^T) \tag{6.2.4}$$

$$\frac{\partial v(A)}{\partial A} = \frac{\sqrt{y_i^T A^T A y_i}}{\sqrt{x_i^T A^T A x_i}} A x_i x_i^T + \frac{\sqrt{x_i^T A^T A x_i}}{\sqrt{y_i^T A^T A y_i}} A y_i y_i^T \tag{6.2.5}$$

From Eq (6.2.2, 6.2.3, 6.2.4, 6.2.5), the complexity of computing $f(A)$'s gradient is $O(s \times d \times m)$. As a result, the complexity of CSML algorithm is $O(r \times b \times g \times s \times d \times m)$ where r is the number of iterations used to optimise A_0 repeatedly (at line 3 in Algorithm 6.1), b is the number of values of β tested in cross validation process (at line 3b in Algorithm 6.1), g is the number of steps in the Conjugate Gradient method.

6.3 Application to Face Verification

In this section, we show how CSML can be applied to face verification on the LFW dataset in detail. In LFW, there are three available versions: original, funneled, and

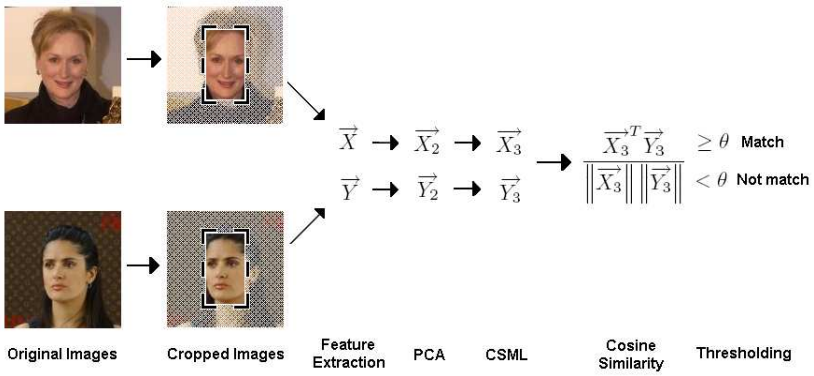

Figure 6.2: The CSML face verification pipeline

aligned. In [81], Wolf et al. showed that the aligned version is better than funneled version at dealing with misalignment. Therefore, we will use the aligned version in our experiments.

6.3.1 Face verification pipeline

The overview of our method is presented in Figure 6.2. First, two original images are cropped to smaller sizes. Next some feature extraction method is used to form feature vectors $(\vec{X}, \vec{Y})$ from the cropped images. These vectors are passed to PCA to get two dimension-reduced vectors $(\vec{X_2}, \vec{Y_2})$. Then CSML is used to transform $(\vec{X_2}, \vec{Y_2})$ to $(\vec{X_3}, \vec{Y_3})$ in the final subspace. Cosine similarity between $\vec{X_3}$ and $\vec{Y_3}$ is the similarity score between two faces. Finally, this score is thresholded to determine whether two faces are the same or not. The optimal threshold θ is estimated from the training set. Specifically, θ is set so that False Acceptance Rate equals to False Rejection Rate. Each step will be discussed in detail.

6.3.1.1 Preprocessing

The original size of each image is 250×250 pixels. At the preprocessing step, we simply crop the image to remove the background, leaving a 80×150 face image. The next step after preprocessing is to extract features from the image.

6.3.1.2 Feature Extraction

To test the robustness of our method to different types of features, we carry out experiments on three facial descriptors: Intensity, Local Binary Patterns and Gabor Wavelets.

Intensity is the simplest feature extraction method. The feature vector is formed by concatenating all the pixels. The length of the feature vector is $12,000$ ($= 80 \times 150$).

Local Binary Patterns (LBP) was first applied for face recognition in [21] with very promising results. In our experiments, the face is divided into non-overlapping 10×10 blocks and LBP histograms are extracted in all blocks to form the feature vector whose length is $7,080$ ($= 8 \times 15 \times 59$).

Gabor Wavelets [18, 80] with 5 scales and 8 orientations are convoluted at different pixels selected uniformly with the downsampling rate of 10×10. The length of the feature vector is $4,800$ ($= 5 \times 8 \times 8 \times 15$) .

6.3.1.3 Dimension Reduction

Before applying any learning method, we use PCA to reduce the dimension of the original feature vector to a more tractable number. A thousand faces from training data (different for each fold) are used to create the covariance matrix in PCA. We notice

in our experiments that the specific value of the reduced dimension after applying PCA does not affect the accuracy significantly.

6.3.1.4 Feature Combination

We can further improve the accuracy by combining different types of features. Features can be combined at the feature extraction step [86, 24] or at the verification step. Here we combine features at the verification step using SVM [8]. Applying CSML to each type of feature produces a similarity score. These scores form a vector which is passed to SVM for verification.

6.3.2 How to choose A_0 in CSML?

Because CSML improves the accuracy of A_0, it is a good idea to choose matrix A_0 which performs well by itself. There are published papers concluding that Whitened PCA with Cosine Similarity can achieve very good performance [26, 27]. We have also discussed the advantages of using Whitened PCA thoroughly in chapter 3. Therefore, the whitening matrix is proposed to be used as A_0. Since the dimension is reduced from m to d, the whitening matrix is in the rectangular form as follows:

$$A_{WPCA} = \begin{bmatrix} \lambda_1^{-\frac{1}{2}} & 0 & \cdots & 0 & 0 & 0 \\ 0 & \lambda_2^{-\frac{1}{2}} & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & & \vdots & 0 & 0 \\ 0 & 0 & \cdots & \lambda_d^{-\frac{1}{2}} & 0 & 0 \end{bmatrix} \in R^{d \times m}$$

where $\lambda_1, \lambda_2, ..., \lambda_d$ are the first d largest eigen-values of the covariance matrix com-

puted in the PCA step.

To compare, we tried two different matrices: non-whitening PCA and Random Projection.

$$A_{PCA} = \begin{bmatrix} 1 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & & \vdots & 0 & 0 \\ 0 & 0 & \cdots & 1 & 0 & 0 \end{bmatrix} \in R^{d \times m}$$

$$A_{RP} = random\ matrix \in R^{d \times m}$$

6.4 Experimental Results

To evaluate performance on View 2 of the LFW dataset, we used five of the nine training splits as training samples and the remaining four as validation samples in CSML algorithm (more about the LFW protocol in [1]). These validation samples are also used for training the SVM. All results presented here are produced using the parameters: $m = 500$ and $d = 200$ where m is the dimension of the data after applying PCA and d is the dimension of the data after applying CSML. In this section, the results of two experiments will be presented.

In this first experiment, we will show how much CSML improves over three cases of A_0: Random Projection, PCA, and Whitened PCA. We call the transformation matrices of these A_{RP}, A_{PCA}, and A_{WPCA} respectively. Here the original intensity is used as the feature extraction method. As shown in Table 6.1, A_{CSML} consistently performs better

	A_{RP}	A_{PCA}	A_{WPCA}
A_0	0.5752 ± 0.0057	0.6762 ± 0.0075	0.7322 ± 0.0037
A_{CSML}	0.673 ± 0.0095	0.7112 ± 0.0083	0.7865 ± 0.0039

Table 6.1: A_{CSML} and A_0 performance comparison

than A_0 about $5 - 10\%$.

In the second experiment, we will show how much CSML improves over Cosine Similarity in the original space and over Whitened PCA with three types of features: Intensity (IN), Local Binary Patterns (LBP), and Gabor Wavelets (GABOR). Each type of feature is tested with the original feature vector or the square root of the feature vector [38, 81, 8].

		Cosine	Whitened PCA	CSML
IN	original	0.6567 ± 0.0071	0.7322 ± 0.0037	0.7865 ± 0.0039
	sqrt	0.6485 ± 0.0088	0.7243 ± 0.0038	0.7887 ± 0.0052
LBP	original	0.7027 ± 0.0036	0.7712 ± 0.0044	0.8295 ± 0.0052
	sqrt	0.6977 ± 0.0047	0.7937 ± 0.0034	$\mathbf{0.8557 \pm 0.0052}$
GABOR	original	0.672 ± 0.0053	0.7558 ± 0.0052	0.8238 ± 0.0021
	sqrt	0.6942 ± 0.0072	0.7698 ± 0.0056	0.8358 ± 0.0058
Feature Combination				$\mathbf{0.88 \pm 0.0037}$

Table 6.2: The improvements of CSML over Cosine Similarity and Whitened PCA

As shown in Table 6.2, CSML improves about 5% over Whitened PCA and about $10 - 15\%$ over Cosine Similarity. LBP seems to perform better than Intensity and Gabor

Wavelets. Using square root of the feature vector improves the accuracy about $2-3\%$ in most cases. The highest accuracy which can be achieved from a single type of feature is 0.8557 ± 0.0052 using CSML with the square root of the LBP feature. The accuracy which can be achieved by combining 6 scores corresponding to 6 different features (in the rightmost column in Table 6.2) is 0.88 ± 0.0038. This is better than the current state of the art result reported in [81]. For comparison purpose, the ROC curves of our method and others are depicted in Figure 6.3. Since there are nearly 20 curves in the figure, only our curves are colorized as non-black to make it easier to compare our results with others. Complete benchmark results can be found in Table 6.3.

6.5 Discussion and Conclusion

A novel method for learning a distance metric based on Cosine Similarity has been introduced. The use of Cosine Similarity allows us to form a simple but effective objective function, which leads to a fast gradient-based optimisation algorithm. Another important property of our method is that in theory the learned matrix cannot perform worse than the regularised matrix. In practice, it performs considerably better in most cases. Our method is tested on the LFW dataset and achieved highest accuracy in the literature. Although initially CSML was designed for face verification, it has a wide range of applications, which we plan to explore in future work (more about this in chapter 8).

Having presented all theoretical contributions of the thesis, we will next discuss pracitical issues in building a full face verification system. This is also our experience participating in ICPR 2010 Face Verification contest.

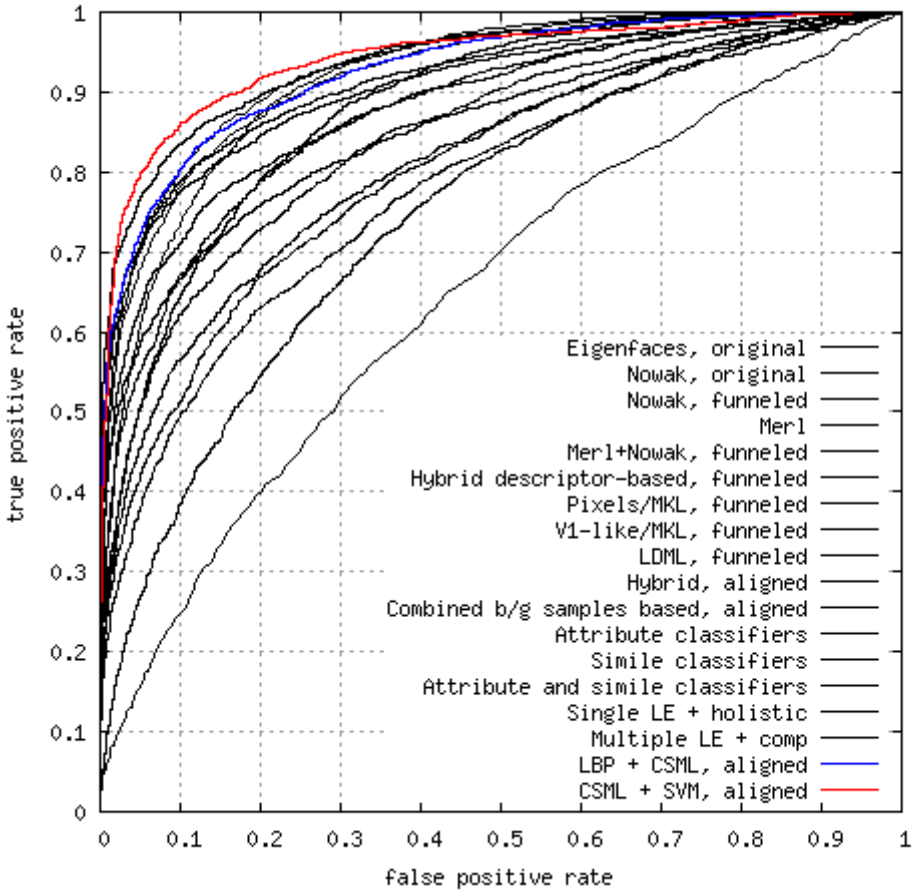

Figure 6.3: ROC curves averaged over 10 folds of View 2

Method	$\mu \pm S_E$
Eigenfaces [29], original	0.6002 ± 0.0079
Nowak [62], original	0.7245 ± 0.0040
Nowak [62], funneled [87]	0.7393 ± 0.0049
MERL [63]	0.7052 ± 0.0060
MERL+Nowak [63], funneled	0.7618 ± 0.0058
Hybrid descriptor-based [8], funneled	0.7847 ± 0.0051
Pixels/MKL [83]	0.6822 ± 0.0041
V1-like/MKL [83]	0.7935 ± 0.0055
LDML, funneled [38]	0.7927 ± 0.0060
Hybrid, aligned [84]	0.8398 ± 0.0035
Combined b/g samples based methods, aligned [81]	0.8683 ± 0.0034
Attribute classifiers [64]	0.8362 ± 0.0158
Simile classifiers [64]	0.8414 ± 0.0131
Attribute and Simile classifiers [64]	0.8529 ± 0.0123
Single LE + holistic [88]	0.8122 ± 0.0053
Multiple LE + comp[88]	0.8445 ± 0.0046
LBP + CSML, aligned [89]	$\mathbf{0.8557 \pm 0.0052}$
CSML + SVM, aligned[89]	$\mathbf{0.8800 \pm 0.0037}$

Table 6.3: Complete benchmark results on the LFW dataset (Image-Restricted)

CHAPTER 7

A Complete Face Verification System

By early 2010, The MOBIO consortium organised a contest for the International Conference on Pattern Recognition (ICPR) 2010. The contest focused on evaluating the performance of uni-modal face and speaker verification techniques in the context of a mobile environment, thus offering challenging recording conditions (adverse illumination, noisy background). We participated with our face verification (only) system. Similar to Face Verification Contest at ICPR 2004 [90], this contest was video-based. However, it was more challenging because participants had to work with raw videos as input instead of normalised face images. In other words, all submissions had to be full systems. Developing a full face verification system is not an easy task. It involves all steps in the face recognition pipeline: face detection, normalisation, feature extraction, feature selection, subspace learning, recognition, and classifier fusion (chapter 2). While previous chapters focus on theoretical issues, this chapter focuses more on practical issues. The objective is to illustrate how a full face verification system can be built in practice and how challenging the task is.

7.1 Our Solution

Our approach followed the general framework discussed in chapter 2. The only additional step is that we need to decide whether we use all frames extracted from a video or only a subset before we can detect faces. In our approach, we did not use all frames but only selected a small number of representative frames. We used OpenCV 2.0 for face detection. We developed a custom algorithm to locate eyes used in the normalisation step. For feature extraction, we used 4 facial descriptors: Raw Image Intensity, Local Binary Patterns [21], Gabor Filters, and Local Gabor Binary Patterns [24, 26]. We used 2 subspace learning methods: Whitened PCA (discussed in chapter 3) and One-shot LDA [8]. At the classifier fusion step, we used Radial Basis Function SVM to combine all features and subspace methods to create the final verifier. The very first step is enrolment, i.e. extracting representative frames from a video.

7.1.1 Enrolment

Although this step looks simple at first, it is actually very hard to do it well. Figure 7.1 presents a number of frames extracted randomly from a video. Since people in videos were not asked to pay attention to the view point, in some cases their faces could not be fully captured by the camera (as frames in the first and second rows). This often happens at the beginning of the recording. The good news is that people might move their heads around during the recording and at some stage reveal their faces fully to the camera (as frames in the last row). The difficult part is how to detect those moments.

We solved this problem by using Locally Linear Embedding (LLE) (chapter 2) [32] to select only five most representative frames in a video [31]. First, we applied LLE for all

Figure 7.1: Illustration of the enrolment step

frames to transform data to two-dimensional space. Then we used 5-means clustering algorithm to select the best 5 frames. Figure 7.2 illustrates these two steps. We found that this solution does not work one hundred percent of the time but still much better than selecting frames randomly. After all, the objective is to select 5 frames which include as full as possible face regions. These 5 frames are passed to the face detector in the next step.

7.1.2 Face Detection

In fact, the objective of this step is not only to detect the face but also detect the eyes. The reason we would like to also detect the eyes is that the eye locations are very important in the face normalisation step that follows. Knowing the eye locations makes

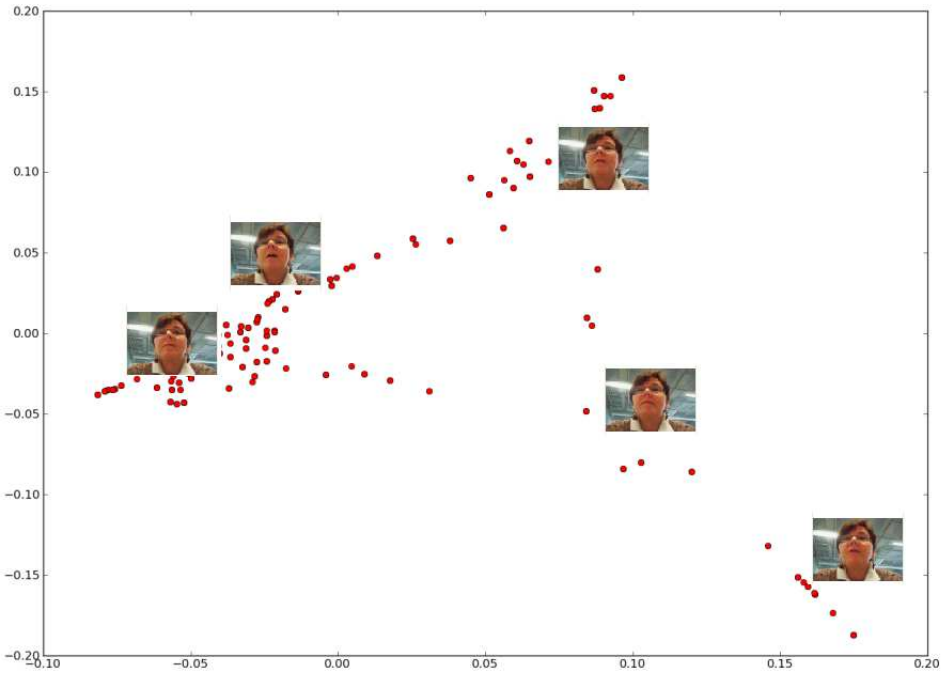

Figure 7.2: Using LLE to select five most representative frames

the normalisation step much more accurate.

To detect the face, we used OpenCV's Haar Feature-based Cascade Classifier [2] with the following parameters: cvHaarDetectObjects(gray, cascade, storage, scale_factor=1.1, min_neighbour=3, flags=0, min_size=cvsize(150, 150)). The min_size was set to 150 empirically. In cases where OpenCV returns more than one face regions (we found this happens very often), we need some algorithm to filter out all but the correct one. The only correct face region has to go through all three following filters.

First, PCA is used to learn the face subspace and all regions which are far enough from that subspace have been removed. Second, among the rest, all regions containing too high or too low percent of skin color have been removed. Third, among the rest, we select the first region which contains two eyes. Our eye locator works as follows.

Eye region is defined as the upper half of the face image and eye detection works on the left and right half of the eye region respectively for the left and right eyes. First, it detects rotationally symmetric (circular) objects using generalised symmetry transform. Edges are detected using Canny edge detection and all edge points are paired to vote the midpoint of their connection for potential symmetry centers with symmetry scores. The symmetry scores are contributed by the symmetry and magnitude of image gradients at the pair of edge points. An expected size of eyes or irises is also compared with the actual distance between the pair of edge points to scale the score. The original image is therefore transformed to a symmetry map and the point in the map with the maximal symmetry score is selected as the position of eye candidates. Next a circular shape template for iris is used to locate the iris in the neighbourhood of eye candidates by an exhaustive search or random search. With properly defined energies based on the edge map, the symmetry map and grayscale values of the original image, the

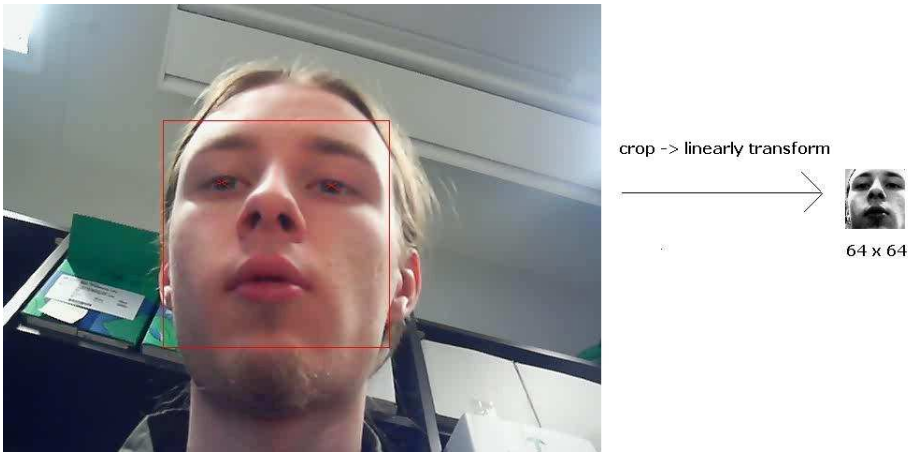

Figure 7.3: Illustration of the challenging of enrolment

search explores the iris state space to find the state where the energy is minimised. The detector finally outputs the coordinates and size (radius) of the iris.

The locations of the face and two eyes have been passed to the next step: normalisation.

7.1.3 Normalisation

The entire step is illustrated in Figure 7.3. First, the face region is cropped and converted to grayscale. Then this cropped grayscale region went through a linear transformation so that after the transformation, two eyes are located at specific locations predefined by a face template (that is why the locations of eyes are so important). The transformation involves a combination of scaling, translation and rotation. Figure 7.4 presents a number of face images after normalisation. As we can see, there are some partially covered faces (like the one at the bottom left corner). This is because of all errors aggregated in the previous steps, especially, eye detection.

The next step is feature extraction.

Figure 7.4: The MOBIO sample faces

7.1.4 Feature Extraction

We used 4 different features: Intensity values (IN), Local Binary Patterns (LBP), Gabor Filters (Gabor), and Local Gabor Binary Patterns (LGBP) (chapter 2).

Intensity value is simply the grey intensity of each pixel. The length of the feature vector is the number of pixels, 4096 (64×64).

LBP was first applied for face recognition in [21] with very promising results. In our implementation, the face is divided into non-overlapping 8×8 blocks and LBP histograms are extracted in all blocks to form the feature vector whose length is $3,776$ ($59 \times 8 \times 8$).

Gabor Filter with 5 scales and 8 orientations are convoluted at different pixels selected uniformly with the downsampling rate of 4×4. The length of the feature vector is $10,240$ ($5 \times 8 \times 16 \times 16$).

The last type of feature is LGBP [24, 25, 26]. There are total of $151,040$ ($5 \times 8 \times 59 \times 16 \times 16$) LGBP features. All features are sorted in descending order of their variances.

The first 15,000 features are selected to form the feature vector.

The feature vector is now passed to the next step.

7.1.5 Dimension Reduction

We used Whitened PCA and One-shot learning with LDA. Since Whitened PCA has been discussed extensively throughout the thesis, we only describe One-shot learning in detail here.

One-shot learning [8] is used to compute the similarity between two input faces. That similarity score is called One-Shot Similarity (OSS). Given two vectors I and J their OSS score is computed by making use of a training set of background sample vectors A. This set contains examples of unlabelled items which are chosen not to belong to the same class as neither I nor J. The similarity of I and J is then computed as follows. First, a discriminative model is learned with A as a set of negative examples, and I as a single positive example. Next this model is used to classify the vector, J, and acquire a confidence score. A second such score is then acquired by repeating the same process with the roles of I and J switched. The final OSS score is the average of these two scores.

In fact, the OSS score is meta-similarities which can be used to work with almost any discriminative learning algorithm. In our experiments, we used the Linear Discriminant Analysis (LDA) (chapter 2) as the underlying classifier. In this special case, similarities based on LDA can be efficiently computed by exploiting two facts. First, the set A of negative samples is used repeatedly. Second, the positive class, which contains just one or two elements, contributes either nothing or a rank-one matrix to the within

class covariance matrix. Also, the fact that the problem is two-class classifier can be used to simplify the LDA algorithm.

Let $p_i \in R^d$, $i = 1, 2, \ldots, m_1$ be a set of positive training samples, and let $n_i \in R^d$, $i = 1, 2, \ldots, m_2$ be a set of negative training samples. Let μ be the average of all samples and μ_p (μ_n) be the average of samples in the positive (negative) training set. In LDA, we need to compute two important matrices: between-class scatter S_B and within-class scatter S_W (as in chapter 2). The LDA algorithm computes a projection v which maximises the quotient:

$$v = argmax_v \frac{v^T S_B v}{v^T S_W v}$$

In the two class case, v can be simply computed as:

$$v = \frac{S_W^{-1}(\mu_p - \mu_n)}{\left\| S_W^{-1}(\mu_p - \mu_n) \right\|}$$

One potential problem is that the S_W matrix can be singular. We solved this by applying PCA to reduce the dimension to a small number which guarantees S_W be nonsingular. Once v has been computed, the classification of a new sample $x \in R^d$ is given by the sign of $v^T x - v_0$, where v_0 is the bias term (see below). By exploiting the fact that the negative set is fixed and the positive set contains a single sample, the LDA-based OSS between samples I and J given the supplementary set A becomes:

$$\frac{(I - \mu_A)^T S_W^{-1}(J - \frac{I+\mu_A}{2})}{\left\| S_W^{-1}(I - \mu_A) \right\|} + \frac{(J - \mu_A)^T S_W^{-1}(I - \frac{J+\mu_A}{2})}{\left\| S_W^{-1}(J - \mu_A) \right\|}$$

How to come up with the final verification from these scores will be discussed next.

7.1.6 Authentication and Classifier Fusion

Whitened PCA and One-shot LDA (OS-LDA) [8] are used to compute the similarity between two input faces. Four features and two subspace methods form a total of 8 similarity scores which can be considered as a $8-D$ vector. This $8-D$ vector is passed to RBF SVM for verification (chapter 2).

RBF-SVM parameters (c and γ) are trained by cross validation technique using LIBSVM library [91]. Training and testing sets are split so that they do not share any common subject. In other words, any subject appears in either training or testing set, exclusively. If training set and testing sets share common subjects, overfitting happens. We made this mistake in our actual submission and it will be discussed more in the next section. Then the final score is normalised to a number between 0 and 1 representing the probability of two input faces matching. Next we will present the experimental results.

7.2 Experimental Results

As mentioned, our approach was tested extensively with four feature extractions methods: Intensity, LBP, Gabor Wavelets, LGBP, and two subspace learning methods: Whitened PCA and OS-LDA. According to the MOBIO protocol 58, the evaluation set is split into male and female subsets. Detail results are presented in Table 7.2 (for female) and Table 7.1 (for male). In these tables, SVM 1 and SVM 2 are algorithms used to combine multiple features and learning methods. SVM 1 is the flawed version and SVM 2 is the correct one.

SVM 1 is the version which we submitted in the real contest. In that version, we made

Classifier\Feature	IN	LBP	GABOR	LGBP
Whitened PCA	25.7/28.1	22.4/23.0	17.0/19.8	20.0/21.4
OS-LDA	23.9/27.4	21.9/28.1	15.4/19.4	23.4/27.1
SVM 1	4.7/29.8			
SVM 2	11.5/14.2			

Table 7.1: Results for male data (HTER for DEV/TEST in %).

a mistake which caused SVM 1 to be overfitted. It is because of the way parameters in SVM 1 have been trained. In the cross validation step, the same numbers of positive samples and negative samples have been used. All samples have been split into two disjoint sets for training and testing. The problem is that the images from the same subject may be assigned to both training and testing sets and the faces from a subject in a session look quite similar.

SVM 2 is the correct version after we fixed the bug in SVM 1. In SVM 2, all samples from a subject can be assigned to either the training subset or the testing subset. Since SVM 2 does not suffer from overfitting, it achieved much better testing results as shown in both tables. Another minor observation from the results is that Gabor features perform consistently well over all datasets.

Table 7.3 presents the final results of the contest. UON 1 is our flawed submission and UON 2 is the fixed version of UON 1. Our correct version UON 2 achieved quite good result and only ranked behind UNIS and VISIDON. One difference between the systems of UNIS and VISIDON and the rest is the use of commercial software. UNIS and VISIDON used commercial software to detect faces and eyes. In addition, their software was tuned aggressively for a long period of time in practice. For a noisy

Classifier\Feature	IN	LBP	GABOR	LGBP
Whitened PCA	22.5/27.9	22.8/27.7	17.0/24.2	17.9/24.0
OS-LDA	20.8/27.1	22.6/29.8	19.9/21.3	20.8/28.1
SVM 1	8.5/23.9			
SVM 2	13.9/17.1			

Table 7.2: Results for female data (HTER for DEV/TEST in %).

training dataset such as MOBIO, using a more reliable face and eye detector does make a real difference in terms of accuracy. It is likely that our system will be improved further using such commercial software.

7.3 Discussion and Conclusion

In this chapter, we described a full face verification system which was our experience taking part in ICPR 2010 Face Verification contest. We showed that building such a system is quite a challenging task. The difficulty comes from the large number of steps involved. Any step going wrong can affect the performance of the entire system. In other words, in order to have a good system, we have to optimise all steps intensively even seemingly unimportant step such as locating eyes. Also from the experience of building this system, we found that multiple features and multiple learning methods do improve the accuracy considerably.

	MALE	FEMALE	AVERAGE
IDIAP	25.4	24.3	24.9
ITI	16.9	17.8	17.4
NICTA	26.2	21.9	24.1
TEC	31.4	29.8	30.2
UNIS	**11.8**	**14.0**	**12.9**
VISIDON	**10.3**	**14.9**	**12.6**
UON 1	29.8	23.9	26.8
UON 2	**14.2**	**17.1**	**15.6**

Table 7.3: ICPR 2010 Face Verfication Contest Final Result (HTER in %)

CHAPTER 8

Conclusions and Future Works

This thesis addresses three important problems in face recognition: lack of training data, accuracy-speed trade-off and unconstrained environments. The problem of lacking training data can be solved effectively using Whitened PCA. Compact Binary Patterns and Multi-patch classifier together can improve the accuracy significantly with a very small speed overhead. We have proposed two new metric learning methods called Indirect Neighbourhood Component Analysis and Cosine Similarity Metric Learning to deal with the last problem of unconstrained conditions and our methods achieved the best results in literature. Last but not least, we described a full face verification system which was our experience taking part in ICPR 2010 Face Verification competition. This chapter summarises our contributions and suggests directions for future works.

8.1 Summary of Contributions

8.1.1 Whitened Principal Component Analysis

The problem of lacking training data and why a good solution is desirable have been discussed. First, many learning algorithms require a large amount of training data to perform well. The number of samples per class should be about ten times as many as the dimension of feature vector. Unfortunately, that amount of data is impossible to collect in many real world applications. Even worse, lacking training data causes overfitting problem which again leads to low recognition rate. Second, small amount of training data saves storage cost and requires less time to train the model. Our proposed method, Whitened PCA, works well under the worst case scenario: only a single sample per person.

8.1.2 Compact Binary Patterns and Multi-patch Classifier

In face recognition, both accuracy and speed are important. Fast software is of no use if it returns incorrect answers. An absolutely correct solution is impractical if it takes days to process a face image. In reality, there is often a trade-off between accuracy and speed. Two methods to improve accuracy without compromising speed noticeably have been proposed. First, Compact Binary Patterns (CBP) is a generalised version of Local Binary Patterns (LBP). It improves LBP by searching for the most compact binary patterns among all possible patterns. As shown in chapter 4, CBP is both more accurate and faster to compute than LBP. Second, multi-patch classifier is a linear combination of classifiers created from overlapped patches on the face. Linear weights are estimated using Memetic Algorithm (MA) which is a powerful general optimisation technique.

MA converges very fast thus takes little training time. Even better, as shown in chapter 4, we do not use the full ensemble but only a small number of patches to have the same accuracy. The multi-patch classifier is more accurate than the single classifier without being much slower.

8.1.3 Indirect Neighbourhood Component Analysis

Face recognition under unconstrained environments is a very big problem. Indirect Neighbourhood Component Analysis (INCA) is our first attempt to solve it. INCA is based on two recently introduced ideas: One-shot learning and Neighbourhood Component Analysis (NCA). In One-shot learning, two testing faces are compared indirectly through faces in a negative set. On the one hand, since one-shot learning only needs a set of additional negative samples, it makes no use of training data. On the other hand, NCA requires labelled training samples which are not available in the LFW dataset. In INCA, we extend NCA with an additional negative set of face images so that class labels are not required for training. The second contribution is that we propose a variant of the kNN algorithm particularly designed for verification. We obtain very promising results on the LFW dataset.

8.1.4 Cosine Similarity Metric Learning

Our second attempt to solve unconstrained face recognition is Cosine Similarity Metric Learning (CSML). Unlike all current metric learning methods using the Euclidean distance as the metric in the transformed subspace, CSML uses the Cosine Similarity. Note that this is not a matter of choosing the Euclidean distance or the Cosine Similarity af-

ter PCA. The difference is more subtle. The Cosine Similarity has a special property that makes it suitable for metric learning, i.e. the resulting similarity measure is always within the range of -1 and $+1$. Because of this property, the objective function can be represented as the sum of all samples. This is much simpler than the objective functions in methods using the Euclidean distance such as Neighbourhood Component Analysis, Large Margin Nearest Neighbour, and Information-Theoretic Metric Learning. Moreover, CSML has the ability to improve a predefined metric considerably. When the predefined metric is set to the rectangular form of Whitened PCA matrix, CSML produces the best results on the LFW dataset in the literature.

8.1.5 Complete Face Verification System

Practically speaking, building a complete face recognition system is highly challenging. The reason is that it involves many steps, all of which need to be optimised in order to have a well working system. Any poorly produced step can decrease the performance of the entire system. In this thesis, we share our experience of building such a system in the process of taking part in the ICPR 2010 Face Verification contest. This system is not about developing new techniques or methods but about how to selectively put all existing techniques together to build a complete working system. From our experience, we found a number of interesting observations. First, face detection and normalisation including eye detection are important to the final accuracy. Since these are two first steps in the chain (see chapter 2), if they are not produced properly then it is not enough to recover no matter how good the later steps are. Second, combining multiple features with SVM does improve accuracy. Third, combining multiple subspace learning methods also improves accuracy. This agrees with what we have discussed about

classifier fusion in chapter 4.

8.2 Future Works

8.2.1 Alternative of Local Gabor Binary Patterns

In terms of raw accuracy, Local Gabor Binary Patterns is currently the best feature extraction method in face recognition. It combines successfully the best features from two other state-of-the-art methods: Local Binary Patterns and Gabor Wavelets. While Local Binary Patterns is very good at dealing with local changes and rotation, Gabor Wavelets is robust to noise, distortion, illumination, and scale. In most cases, Local Gabor Binary Patterns performs better than both Local Binary Patterns and Gabor Wavelets. The problem with LGBP is its extremely slow speed. We have to convolute Gabor Filters with the entire image and this is 40 times slower than Local Binary Patterns. One possible solution is to replace Gabor Wavelets with a new type of feature which is less redundant. The Dual-Tree Complex Wavelet Transform [92] shares many common properties with Gabor Wavelets but runs much faster thus will be a very promising candidate. We plan to find a way to combine Dual-Tree Complex Wavelets with Local Binary Patterns.

8.2.2 Cosine Similarity Metric Learning for Face Identification

Cosine Similarity Metric Learning (CSML) was originally designed for face verification. Although some modifications will be needed, applying CSML to face identification should not be any harder. The first modification is to change the way the cross validation error is evaluated. In face identification, the cross validation error is sim-

ply the identification rate. The second modification is required if label of each training sample is provided. In that case, we need to convert the training data to the form of positive/negative pairs. We plan to apply CSML to databases which are specially designed for face identification such as the FERET and ORL datasets.

8.2.3 Cosine Similarity Metric Learning as a General Metric Learning Framework

Although CSML is designed to solve the problem of face recognition, it is not limited to face recognition by any means. Many researchers around the world have asked us to publish a general metric learning framework based on Cosine Similarity Metric Learning. We think this is a great idea. Then after making CSML a general framework, we can directly compare it with all other metric learning algorithms on general machine learning datasets. It will be very interesting to see how CSML performs compared with other state-of-the-art metric learning methods such as Large Margin Nearest Neighbour or Information-Theoretic Metric Learning in general. We plan to carry out extensive experiments for comparison.

Bibliography

[1] G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical Report 07-49, University of Massachusetts, Amherst, October 2007.

[2] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the Computer Vision and Pattern Recognition*, 2001.

[3] T. Ojala, M. Pietikäinen, and T. Mäenpää. Multiresolution Gray-Scale and Rotation Invariant Texture Classification with Local Binary Patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 971–987, 2002.

[4] K. Weinberger, J. Blitzer, and L. Saul. Distance metric learning for large margin nearest neighbor classification. *Advances in Neural Information Processing Systems*, 18:1473–1480, 2006.

[5] V. Vapnik and A. Lerner. Pattern recognition using generalized portrait

method. *Automation and Remote Control*, 24:774–780, 1963.

[6] A. K. Jain and B. Chandrasekaran. Dimensionality and sample size considerations in pattern recognition practice. *Pattern Recognition*, 1982.

[7] Y. Adini, Y. Moses, and S. Ullman. Face Recognition: The Problem of Compensating for Changes in Illumination Direction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 721–732, 1997.

[8] L. Wolf, T. Hassner, and Y. Taigman. Descriptor based methods in the wild. In *Real-Life Images workshop at the European Conference on Computer Vision (ECCV)*, October 2008.

[9] J. Goldberger, S. Roweis, G. Hinton, and R. Salakhutdinov. Neighborhood component analysis. In *The Neural Information Processing Systems (NIPS)*, 2004.

[10] http://vis-www.cs.umass.edu/lfw/results.html. LFW benchmark results visited by April 30, 2011.

[11] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference and Prediction*. Springer; 1st ed. 2001. Corr. 3rd printing edition, 2003.

[12] T. K. Ho, J. J. Hull, and S. N. Srihari. Decision combination in multiple classifier systems. *IEEE Trans. Pattern Anal. Mach. Intell.*, 16(1):66–75, 1994. ISSN 0162-8828.

[13] H. Wechsler. *Reliable Face Recognition Methods: System Design, Implementation and Evaluation*. Springer, 2006.

[14] R. Chellappa, C.L. Wilson, S. Sirohey, et al. Human and machine recognition of faces: a survey. *Proceedings of the IEEE*, 83(5):705–740, 1995.

[15] W. Zhao, R. Chellappa, P. J. Phillips, and A. Rosenfeld. Face recognition: A literature survey. *ACM Comput. Surv.*, 35:399–458, December 2003. ISSN 0360-0300.

[16] M. Jones and P. Viola. Face recognition using boosted local features. *Proceedings of the International Conference on Computer Vision*, 2003.

[17] C. P. Papageorgiou, M. Oren, and T. Poggio. A general framework for object detection. In *Proceedings of International Conference on Computer Vision*, pages 555–562, 1998.

[18] J.G. Daugman. Complete Discrete 2D Gabor Transforms by Neural Networks for Image Analysis and Compression. *IEEE Transactions on Acoustics Speech Signal Process*, 36, 1988.

[19] M. Lades, J.C. Vorbruggen, J. Buhmann, J. Lange, C. von der Malsburg, R.P. Wurtz, and W. Konen. Distortion invariant object recognition in the dynamic linkarchitecture. *IEEE Transactions on Computers*, 42(3):300–311, 1993.

[20] L. Wiskott, J.M. Fellous, N. Krüger, and C. von der Malsburg. Face Recognition by Elastic Bunch Graph Matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 775–779, 1997.

[21] T. Ahonen, A. Hadid, and M. Pietikainen. Face Recognition with Local Binary Patterns. *Proceedings of European Conference on Computer Vision*, pages 469–481, 2004.

[22] T. Ojala, M. Pietikäinen, and D. Harwood. A comparative study of texture measures with classification based on featured distributions. *Pattern Recognition*, 29(1):51 – 59, 1996. ISSN 0031-3203.

[23] T. Ahonen, A. Hadid, and M. Pietikäinen. Face Description with Local Binary Patterns: Application to Face Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 2037–2041, 2006.

[24] W. Zhang, S. Shan, W. Gao, X. Chen, and H. Zhang. Local Gabor Binary Pattern Histogram Sequence (LGBPHS): A Novel Non-Statistical Model for Face Representation and Recognition. In *Proceedings of the International Conference on Computer Vision*, pages 786–791, 2005.

[25] S. Shan, W. Zhang, Y. Su, X. Chen, and W. Gao. Ensemble of Piecewise FDA Based on Spatial Histograms of Local (Gabor) Binary Patterns for Face Recognition. In *Proceedings of the 18th International Conference on Pattern Recognition*, pages 606–609, 2006.

[26] H. V. Nguyen, L. Bai, and L. Shen. Local gabor binary pattern whitened pca: A novel approach for face recognition from single image per person. In *Proceedings of The 3rd IAPR/IEEE International Conference on Biometrics*, 2009.

[27] W. Deng, J. Hu, and J. Guo. Gabor-Eigen-Whiten-Cosine: A Robust

Scheme for Face Recognition. *Analysis and Modelling of Faces and Gestures*, 3723:336, 2005.

[28] L. Shen and L. Bai. MutualBoost learning for selecting Gabor features for face recognition. *Pattern Recognition Letters*, 27(15):1758–1767, 2006.

[29] M. Turk and A. Pentland. Face recognition using eigenfaces. In *Computer Vision and Pattern Recognition, 1991. Proceedings CVPR'91., IEEE Computer Society Conference on*, pages 586–591, 1991.

[30] J. B. Tenenbaum, V. Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, December 2000. ISSN 00368075.

[31] A. Hadid and M. Pietikäinen. Selecting models from videos for appearance-based face recognition. In *Proceedings of the 16th International Conference on Pattern Recognition*, pages 304–308, 2004.

[32] S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *SCIENCE*, 290:2323–2326, 2000.

[33] W. Zhao, R. Chellappa, and A. Krishnaswamy. Discriminant analysis of principal components for face recognition. In *Automatic Face and Gesture Recognition, 1998. Proceedings. Third IEEE International Conference on*, pages 336–341, 1998.

[34] K. Fukunaga. *Introduction to statistical pattern recognition*. Academic Press, 1991.

[35] D.L. Swets and J.J. Weng. Using Discriminant Eigenfeatures for Image Retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 831–836, 1996.

[36] C. Liu and H. Wechsler. Enhanced Fisher Linear Discriminant Models for Face Recognition. In *Proceedings of International Conference on Pattern Recognition*, volume 14, pages 1368–1372, 1998.

[37] J. V. Davis, B. Kulis, P. Jain, S. Sra, and I. S. Dhillon. Information-theoretic metric learning. In *Proceedings of the 24th international conference on Machine learning*, pages 209–216, 2007. ISBN 978-1-59593-793-3.

[38] M. Guillaumin, J. Verbeek, and C. Schmid. Is that you? metric learning approaches for face identification. In *Proceedings of International Conference on Computer Vision*, pages 498–505, sep 2009.

[39] H. Gupta, A. K. Agrawal, T. Pruthi, C. Shekhar, and R. Chellappa. An experimental evaluation of linear and kernel-based methods for face recognition. In *Proceedings of the Sixth IEEE Workshop on Applications of Computer Vision*, page 13, 2002. ISBN 0-7695-1858-3.

[40] E. Osuna, R. Freund, and F. Girosi. Training support vector machines: an application to face detection. *Proceedings of Computer Vision and Pattern Recognition*, page 130, 1997. ISSN 1063-6919.

[41] P. J. Phillips. Support vector machines applied to face recognition. In *Advances in Neural Information Processing Systems 11*, pages 803–809. MIT Press, 1999.

[42] B. Moghaddam and M.H. Yang. Gender classification with support vector machines. In *Proceedings of International Conference on Automatic Face and Gesture Recognition*, pages 306–311, 2000.

[43] B. Schölkopf, S. Mika, C. J. C. Burges, P. Knirsch, K. Müller, G. Rätsch, and A. J. Smola. Input space vs. feature space in kernel-based methods. IEEE Transactions on Neural Networks, 1999.

[44] B. Scholkopf, A. Smola, and K.R. Muller. Nonlinear Component Analysis as a Kernel Eigenvalue Problem. *Neural Computation*, 10(5):1299–1319, 1998.

[45] G. Baudat and F. Anouar. Generalized discriminant analysis using a kernel approach. Neural Computation, 2000.

[46] K.I. Kim, K.C. Jung, and H.J. Kim. Face recognition using kernel principal component analysis. *Signal Processing Letters*, 9(2):40–42, 2002.

[47] D. Black and R. A. Newing. *The Theory of Committees and Elections*. Cambridge University Press, 1958, 1963.

[48] J. Zou, Q. Ji, and G. Nagy. A comparative study of local matching approach for face recognition. *IEEE Transactions on Image Processing*, 16:2617–2628, 2007.

[49] T. P. Hettmansperger. *Statistical Inference Based on Ranks*. 1984.

[50] A. Agresti. *Categorical Data Analysis*. 1990.

[51] D. R. Cox and E. J. Snell, editors. *Analysis of Binary Data 2nd edition*. Chapman & Hall/CRC, 1989.

[52] B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*, pages 144–152. ACM Press, 1992.

[53] C. Cortes and V. Vapnik. Support-vector networks. In *Machine Learning*, pages 273–297, 1995.

[54] K. Jonsson, J. Kittler, Y.P. Li, and J. Matas. Support vector machines for face authentication. Image and Vision Computing, 1999.

[55] P.J. Phillips, H. Wechsler, J. Huang, and P.J. Rauss. The FERET database and evaluation procedure for face-recognition algorithms. *Image and Vision Computing*, 16(5):295–306, 1998.

[56] http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html. The orl database website visited by april 30, 2011.

[57] E. Bailly-Baillire, S. Bengio, F. Bimbot, M. Hamouz, J. Mariethoz, J. Matas, K. Messer, F. Poree, and B. Ruiz. The banca database and evaluation protocol. Proceedings of International Conference on Audio- and Video-Based Biometric Person Authentication, 2003.

[58] C. McCool and S. Marcel. Mobio database for the icpr 2010 face and speech competition. Technical report, Idiap, November 2009.

[59] L. Shen, L Bai, and M. Fairhurst. Gabor wavelets and general discriminant analysis for face identification and verification. *Image and Vision Computing*, 27:1758–1767, 2006.

[60] H. V. Nguyen and L. Bai. Compact binary patterns (cbp) with multiple patch classifiers for fast and accurate face recognition. In *Proceedings of Computational Modeling of Objects Presented in Images: Fundamentals, Methods, and Applications*, pages 187–198, 2010.

[61] J. Ruiz del Solar, R. Verschae, and M. Correa. Recognition of faces in unconstrained environments: A comparative study. *EURASIP Journal on Advances in Signal Processing*, page 19, 2009.

[62] E. Nowak. Learning visual similarity measures for comparing never seen objects. In *Proceedings of Computer Vision and Pattern Recognition*, 2007.

[63] G. B. Huang, M. J. Jones, and E. Learned-Miller. Lfw results using a combined nowak plus merl recognizer. *Faces in Real-Life Images Workshop in European Conference on Computer Vision*, 2008.

[64] N. Kumar, A. C. Berg, P. N. Belhumeur, and S. K. Nayar. Attribute and simile classifiers for face verification. In *Proceedings of International Conference on Computer Vision*, 2009.

[65] S. Raudys and A. K. Jain. Small sample size effects in statistical pattern recognition: Recommendations for practitioners. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(3):252–264, 1991.

[66] A. M. Martinez. Recognizing imprecisely localized, partially occluded and expression variant faces from a single sample per class. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2002.

[67] A. Pentland, T. Starner, N. Etcoff, N. Masoiu, O. Oliyide, and M. Turk. Experiments with eigenfaces. In *Proceedings of Looking at People Workshop, International Joint Conference on Artifical Intelligence*, 1993.

[68] N. Peter, P. João, and J. David. Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 711–720, 1997.

[69] K.K. Sung and T. Poggio. Example-based learning for view-based human face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(1):39–51, 1998.

[70] B. Moghaddam and A. Pentland. Probabilistic visual learning for object representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):696–710, 1997.

[71] C. Liu. Gabor-Based Kernel PCA with Fractional Power Polynomial Models for Face Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 572–581, 2004.

[72] C. Liu and H. Wechsler. Gabor feature based classification using the enhanced fisher lineardiscriminant model for face recognition. *IEEE Transactions on Image Processing*, 11(4):467–476, 2002.

[73] L. Bai and L. Shen. InfoBoost for Selecting Discriminative Gabor Features. *Proceedings of Computer Analysis of Images and Pattern*, 3691:423, 2005.

[74] L. Shen and L. Bai. A SVM Face Recognition Method Based on Optimized Gabor Features. *Proceedings of the 9th international conference on Advances in visual information systems*, 2007.

[75] P. W. Shor. A new proof of cayley's formula for counting labeled trees. *J. Comb. Theory Ser. A*, 71(1):154–158, 1995. ISSN 0097-3165.

[76] Y. Su, S. Shan, X. Chen, and W. Gao. Patch-based gabor fisher classifier for face recognition. In *Proceedings of International Conference on Pattern Recognition*, pages 528–531, 2006.

[77] C. J.C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2:121–167, 1998.

[78] P. Moscato. On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. 1989.

[79] P. J. Phillips, H. Moon, S. A. Rizvi, and P. J. Rauss. The FERET Evaluation Methodology for Face-Recognition Algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1090–1104, 2000.

[80] S. Shan, W. Gao, Y. Chang, B. Cao, and P. Yang. Review the strength of Gabor features for face recognition from the angle of its robustness to mis-alignment. In *Proceedings of the 17th International Conference on Pattern Recognition*, volume 1, 2004.

[81] L. Wolf, T. Hassner, and Y. Taigman. Similarity scores based on background samples. In *Proceedings of Asian Conference on Computer Vision*, pages 88–97, 2009.

[82] A. Angelova, Y. Abu-Mostafa, and P. Perona. Pruning training sets for learning of object categories. In *Proceedings of Conference on Computer Vision and Pattern Recognition*, pages 494–501, 2005. ISBN 0-7695-2372-2.

[83] N. Pinto, J. J. DiCarlo, and D. D. Cox. How far can you get with a modern face recognition test set using only simple features? *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 0:2591–2598, 2009.

[84] Y. Taigman, L. Wolf, and T. Hassner. Multiple one-shots for utilizing class label information. In *The British Machine Vision Conference (BMVC)*, 2009.

[85] M. Butman and J. Goldberger. Face recognition using classification-based linear projections. *EURASIP J. Adv. Signal Process*, 2008:1–7, 2008. ISSN 1110-8657.

[86] X. Tan, B. Triggs, and M. Vision. Fusing Gabor and LBP Feature Sets for Kernel-Based Face Recognition. *Proceedings of Analysis and Modelling of Faces and Gestures*, 4778:235, 2007.

[87] G. B. Huang and V. Jain. Unsupervised joint alignment of complex images. In *Proceedings of International Conference on Computer Vision*, 2007.

[88] Z. Cao, Q. Yin, X. Tang, and J. Sun. Face recognition with learning-based descriptor. In *Proceedings of Computer Vision and Pattern Recognition*, 2010.

[89] H. V. Nguyen and L. Bai. Cosine similarity metric learning for face verification. In *Proceedings of the 10th Asian Conference on Computer Vision*, pages 709–720, 2010.

[90] K. Messer, J. Kittler, M. Sadeghi, M. Hamouz, A. Kostin, F. Cardinaux, S. Marcel, S. Bengio, C. Sanderson, J. Czyz, L. Vandendorpe, C. McCool, S. Lowther, S. Sridharan, V. Chandran, R. P. Palacios, E. Vidal, L. Bai, L. Shen, Y. Wang, C. Yueh-Hsuan, L. Hsien-Chang, H. Yi-Ping, A. Heinrichs, M. Muller, A. Tewes, C. von der Malsburg, R. Wurtz, Z. Wang, F. Xue, Y. Ma, Q. Yang, C. Fang, X. Ding, S. Lucey, R. Goss, H. Schneiderman, N. Poh, and Y. Rodriguez. Face authentication test on the banca database. *International Conference on Pattern Recognition*, 4:523–532, 2004. ISSN 1051-4651.

[91] http://www.csie.ntu.edu.tw/ cjlin/libsvm/. LIBSVM website visited by April 30, 2011.

[92] N. Kingsbury. The dual-tree complex wavelet transform: A new efficient tool for image restoration and enhancement. In *The 9th European Signal Processing Conference*, pages 319–322.