ENSEMBLE BOOSTING IN COMPLEX ENVIRONMENT AND ITS APPLICATIONS IN FACIAL DETECTION AND IDENTIFICATION

LIU JIANG, JIMMY

NATIONAL UNIVERSITY OF SINGAPORE

2003

ENSEMBLE BOOSTING IN COMPLEX ENVIRONMENT AND ITS APPLICATIONS IN FACIAL DETECTION AND IDENTIFICATION

LIU JIANG, JIMMY

A THESIS SUBMITTED

FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

DEPARTMENT OF COMPUTER SCIENCE

NATIONAL UNIVERSITY OF SINGAPORE

2003

Acknowledgements

I wish to thank many people who have in one way or another helped me while writing this dissertation. No amount of acknowledgements is enough for the advice, efforts and sacrifice of these colleagues and friends who in any case never expect any.

My greatest thank goes to my supervisor, Associate Professor Loe Kia Fock. It was his guidance, care and words of encouragement that enabled me to weather bouts of depression during the four years of academic pursuit. I gained inspiration and enlightenment from Prof. Loe's beneficial discussion and knowledge imparted through his lectures and supervision.

Advice and help rendered to me from my friends Associated Professor Chan Kap Luk from NTU, Dr. Jit Biswas from I^2R, Mr. Andrew David Nicholls, Ms. Lok Pei Mei and Mr. James Yeo will be remembered.

Lastly, the moral support and understandings from my wife and members of the family are crucial for the completion of this dissertation.

Table of Contents

List of Figures

List of Tables

Summary

The Adaptive Boosting (AdaBoost) algorithm is generally regarded as the first practical boosting algorithm, which has gained popularity in recent years. At the same time, its limitation in handling the outliers in a complex environment is also noted. We develop a new ensemble boosting algorithm, S-AdaBoost, after reviewing the popular adaptive boosting algorithms and exploring the need to improve upon the outlier handling capability of current ensemble boosting algorithms in the complex environment. The contribution of the S-AdaBoost algorithm is its use of AdaBoost's adaptive distributive weight as a dividing tool to split up the input space into inlier and outlier sub-spaces. Dedicated classifiers are used to handle the inliers and outliers in their corresponding sub-spaces. The results obtained from the dedicated classifiers are then non-linearly combined. Experimental results of tests derived from some benchmark databases show the new algorithm's effectiveness when compared with other leading outlier handling approaches. The S-AdaBoost machine is made up of an AdaBoost divider, an AdaBoost classifier for inliers, a dedicated classifier for outliers, and a non-linear combiner.

Within the confines of a complex airport environment, to demonstrate the effectiveness of the S-AdaBoost algorithm, we develop the S-AdaBoost based FDAO (Face Detection for Airport Operators) and FISA (Face Identification System for Airports) systems. The FDAO system's performance is compared with the leading face detection approaches using the data obtained from both the complex airport environment and some popular face database repositories. The experimental results demonstrate the effectiveness of the S-AdaBoost algorithm on the face detection application in the real world environment. Similar to the FDAO system, the FISA system's performance is compared with the leading face identification approaches using the airport data and the FERET (FacE REcognition Technology) standard dataset. Results obtained are equally promising and convincing, which shows that the S-AdaBoost algorithm is effective in handling the outliers in a complex environment for the purpose of face identification.

Chapter One

Introduction

1.1 Motivation

This thesis reports some research results conducted in the field of ensemble boosting, an active research stream of machine learning theory. The Ensemble Boosting (or boosting) algorithm [Valiant, 1984; Schapire, 1992] is a special machine learning technique, which intelligently integrates some relatively weak learning algorithms to form a stronger collective one in order to boost the ensemble's overall performance. Recent interest in ensemble boosting is partly due to the success of an algorithm called the AdaBoost (Adaptive Boosting) [Freund and Schapire, 1994]. Implementations of this simple algorithm and the positive results obtained by researchers from using it in various applications [Maclin and Opitz, 1997; Schwenk and Bengio, 1997] have since attracted much research attention.

Researchers, while celebrating the success of the AdaBoost algorithm in some applications, also find that the good performance of the AdaBoost algorithm tends to be restricted to the low noise regime, a drawback which limits its use in the often seen complex real world environments. This drawback is inherent in the design of the AdaBoost algorithm, which focuses on the "difficult" patterns instead of the "easy" ones. As noisy patterns or outliers often fall into the category of the "difficult" patterns, the performance of the AdaBoost algorithm can be affected when the number of outlier patterns becomes large.

To overcome this limitation, many enhanced versions of the AdaBoost algorithm have been proposed [Friedman, Hastie and Tibshirani, 1998; Freund, 1999; Freund, 1995; Domingo and Watanabe, 2000; Servedio, 2001; Mason, Bartlett and Baxter, 1998; Rätsch, Onoda and Müller, 2001] with varying success to expand the AdaBoost algorithm's capability dealing with noise.

Motivated by the effectiveness and elegance of the AdaBoost algorithm and the desire to extend the adaptive boosting approach to the complex real world environment, the S-AdaBoost algorithm [Liu and Loe, 2003a], which utilizes the widely used strategy of "divide and conquer" and is effective in handling outliers, will be discussed in this thesis. The S-AdaBoost algorithm's effectiveness is demonstrated by the experimental results conducted on some benchmark databases through comparing with other leading outlier handling approaches. To further demonstrate the effectives of the S-AdaBoost algorithm in the real world environment, Face Detection for Airport Operators (FDAO) [Liu, Loe and Zhang, 2003c] and the Face Identification System for Airports (FISA) [Liu and Loe, 2003b] systems for a real airport complex environment will be discussed. The experimental results from these systems are compared with other leading face detection and face identification approaches, which clearly show the effectiveness of the S-AdaBoost algorithm.

1.2 Contribution

Solving a complex problem by using the widely used strategy of "divide and conquer", we introduce the S-AdaBoost algorithm. Utilizing the characteristic that the AdaBoost algorithm focuses more on the "difficult" patterns than the "easy" patterns after certain rounds of iteration, an AdaBoost algorithm-based dividing mechanism is implemented to divide the input pattern space into two separate spaces (the inlier sub-space and the outlier space). Dedicated two sub-classifiers are then used to handle the two separate sub-spaces. To further demonstrate the S-AdaBoost algorithm's effectiveness, the algorithm is applied to the face detection and the face identification applications in the complex airport environment. The S-AdaBoost algorithm's effectiveness is demonstrated by the experimental results conducted on some benchmark databases through comparing with other leading outlier handling approaches. To further demonstrate the effectives of the S-AdaBoost algorithm in the real world environment, the Face Detection for Airport Operators (FDAO) and the Face Identification System for Airports (FISA) systems based on S-AdaBoost algorithm, are introduced and discussed in this thesis.

The complex environment associated with pattern detection and pattern identity recognition usually implies, but is not limited to the complication of the background and the complication of the conditions of the object patterns to be detected or identified. This includes those variations such as lighting, coloring, occlusion, and shading; whereas the complex condition of the objects may include the differences in positioning, viewing angles, scales, limitation of the data capturing devices and timing. In the face detection and the face identification applications, the complexity comes from three common factors (variation in illumination, expression, pose / viewing angle) as well as aging, make-up, and the presence of facial features such as a beard and glasses etc. In this thesis, the airport environment is chosen as a typical example of the complex environment for testing, as it contains all the above-mentioned complexity.

To summarize, the main contributions of the thesis are:

- Propose the S-AdaBoost algorithm, which innovatively uses the AdaBoost's adaptive distributive weight as a dividing tool to divide the input space into inlier and outlier sub-spaces and to use dedicated classifiers to handle the inliers and outliers in the corresponding spaces before non-linearly combining the results of the dedicated classifiers.

- The S-AdaBoost algorithm's effectiveness is demonstrated by the experimental results conducted on some benchmark databases through comparing with other leading outlier handling approaches. To further demonstrate the effectives of the S-AdaBoost algorithm in the real world environment, two S-AdaBoost algorithm based application systems, FDAO and FISA are developed. Better experimental results are obtained from the two systems comparing with leading face detection and face identification approaches.

1.3 The Structure of the Thesis

The rest of the thesis is structured as follows: Chapter 2 introduces some of the background information needed in the thesis. The widely used strategy of "divide and conquer" is introduced together with its application in ensemble learning; brief introductions of the face detection and the face identification applications, as well as the state of the art methodologies in the fields are mentioned. Chapter 3 describes the ensemble boosting. The popular adaptive boosting method AdaBoost, the AdaBoost algorithm's effectiveness in preventing overfitting and its ineffectiveness in handling outliers are also described. Chapter 4 introduces the new S-AdaBoost algorithm. The input pattern space in the S-AdaBoost algorithm is analyzed followed by proposing the structure of an S-AdaBoost machine; the S-AdaBoost's divider, its classifiers and its combiner are also introduced. Some theory analysis is provided followed by the experimental results of the S-AdaBoost algorithm on some popular benchmark databases. Chapter 5 focuses on the S-AdaBoost algorithm's applications in the domains of the face pattern detection and the face pattern identification in the complex airport environment. The Face Detection for Airport Operators (the FDAO system) and the Face Identification System for Airports (the FISA system) as well as their implementation details are discussed. The experimental results of the two systems obtained from the airport datasets are compared with the results obtained from other leading face detection and face identification approaches on the same airport datasets. Further experiments from all the approaches are also conducted on the benchmark datasets for the face detection and the face identification applications to further prove the S-AdaBoost algorithm's effectiveness in those applications and datasets. Conclusions are drawn in Chapter 6 followed by the bibliography.

Chapter Two
Background

2.1 Ensemble Learning Classification

A complex computational problem can be solved by dividing it into a number of simple computational sub-tasks, followed by conquering the complex computational problem through combining the sub-solutions to the sub-tasks. In the classification context, computational simplicity and efficiency can be achieved by combining the outputs from a number of sub-classifiers, each of which focuses on the partial or the whole input training space [Chakrabarti Soumen, Shourya Roy and Mahesh Soundalgekar, 2002]. The whole structure is sometimes termed as an Ensemble or Committee Machine [Nilsson, 1965].

In the classification scenario, an ensemble learning classifier $\mathbf{\hat{E}}$ can be defined as an aggregated classifier, which is the combination of several individual component classifiers. It can be denoted by:

$$\mathbf{y_i} = \mathbf{\hat{C}}(\mathbf{\hat{w}}_j(\mathbf{x}_i)) \qquad (2.1.1)$$

Where $\mathbf{y_i} \in \mathbf{Y}$, which stands for the output of the ensemble learning classifier $\mathbf{\hat{E}}$;

$\mathbf{\hat{C}}$ is the Combination function;

$\mathbf{\hat{w}}_j$ (**j** takes its value from1 to **J**, which stands for the total number of the individual component classifiers) is the individual component classifier (sometimes it is called the component classifier, the individual classifier or the base classifier);

$\mathbf{x}_i \in \mathbf{X}$ ($\mathbf{i}$ =1 to $\mathbf{I}$, which stands for the total number of the training input patterns) is the input to the particular individual component classifier $\hat{\mathbf{w}}_{j.}$; and

$\{\mathbf{x}_i, \mathbf{y}_i\}$ denotes a specific training pattern pair.

Ensemble classifiers $\hat{\mathbf{E}}$s can be classified into static and dynamic categories depending on how their input patterns $\mathbf{x}_i$s are involved in forming the structure of the classification mechanism.

In a static ensemble classifier $\hat{\mathbf{E}}$ (as shown in Figure 2.1), a particular input pattern $\mathbf{x}_i$ is involved in the training of the individual component classifiers but not directly involved in the formation of the combination function $\hat{\mathbf{C}}$, which means:

$$\hat{\mathbf{C}} = \hat{\mathbf{C}}\ (\hat{\mathbf{w}}_j) \qquad (2.1.2)$$

In a dynamic ensemble classifier $\hat{\mathbf{E}}$ (as shown in Figure 2.2), the particular input pattern $\mathbf{x}_i$ is directly involved in the formation of the combination function $\hat{\mathbf{C}}$, which means:

$$\hat{\mathbf{C}} = \hat{\mathbf{C}}\ (\hat{\mathbf{w}}_j, \mathbf{x}_i) \qquad (2.1.3)$$

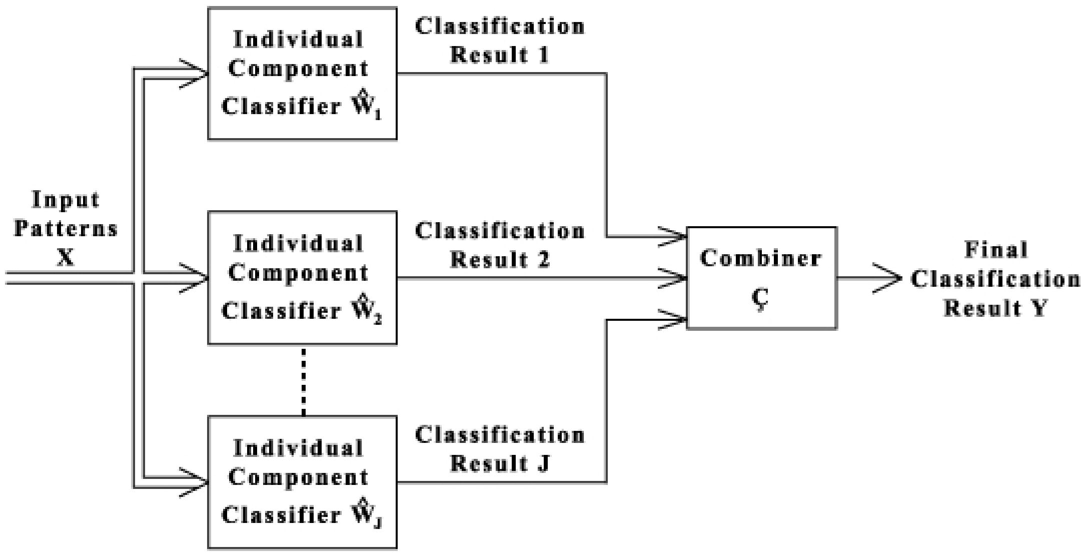
Individual Component Classifier $\hat{W}_1$
Classification Result 1
Input Patterns X
Individual Component Classifier $\hat{W}_2$
Classification Result 2
Combiner Ç
Final Classification Result Y
Individual Component Classifier $\hat{W}_J$
Classification Result J

Figure 2.1 The static ensemble classification mechanism

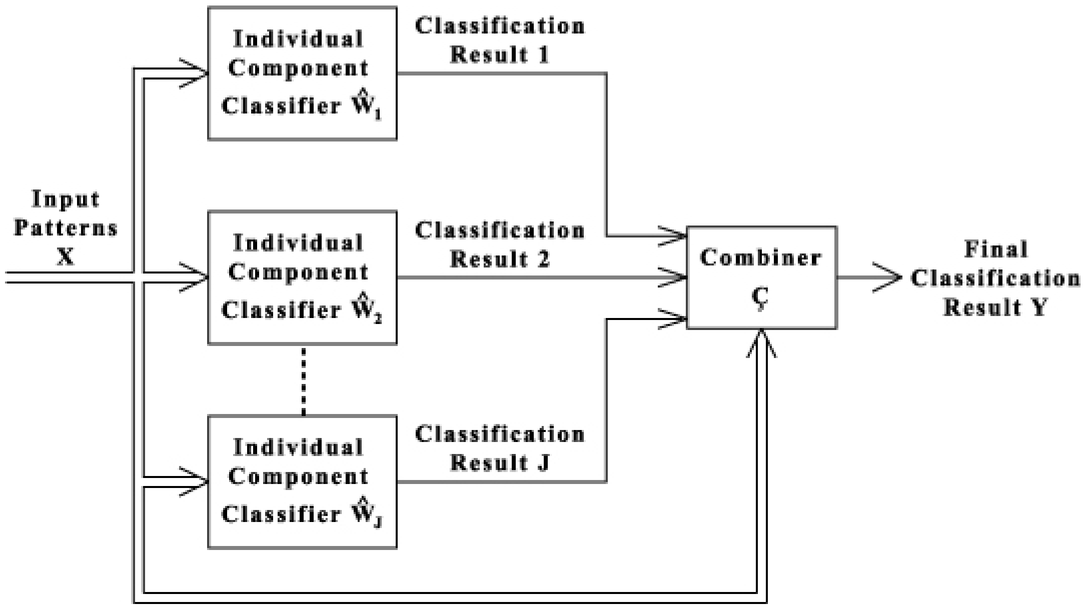
Individual Component Classifier $\hat{W}_1$
Classification Result 1
Input Patterns X
Individual Component Classifier $\hat{W}_2$
Classification Result 2
Combiner Ç
Final Classification Result Y
Individual Component Classifier $\hat{W}_J$
Classification Result J

Figure 2.2 The dynamic ensemble classification mechanism

Two main sub-categories of the static ensemble classifiers **Ê**s are the Ensemble Averaging Classifier **Â** [Wolpert, 1992; Perrone, 1993; Naftaly and Horn, 1997; Hashem, 1997] and the Ensemble Boosting (or Boosting) Classifier **B** [Schapire, 1990]. Outputs of the individual component classifiers $\hat{\mathbf{w}}_i$s are linearly combined by the combiner **Ĉ** to generate the final classification result in an ensemble averaging classifier **Â**. The weak individual component classifiers $\hat{\mathbf{w}}_i$s are boosted during the training process to achieve the final good performance in a boosting classifier **B**. The main difference between the two categories of classifiers is the way that the individual component classifiers $\hat{\mathbf{w}}_i$s are trained in the classifiers. In an ensemble averaging classifier $\hat{\mathbf{A}}$, all of the individual component classifiers $\hat{\mathbf{w}}_i$s are trained on the same training pattern pair set $\{\mathbf{X_i}, \mathbf{Y_i}\}$, even though they may differ from each other in choosing the initial training network parameter settings among the individual component classifiers $\hat{\mathbf{w}}_i$s. Whereas in the ensemble boosting classifier **B**, the individual component classifiers $\hat{\mathbf{w}}_i$s are trained on the entirely different distributions of the training pattern pair set $\{\mathbf{X_i}, \mathbf{Y_i}\}$. Boosting or Ensemble Boosting, which will be discussed in more detail in the following sections and chapters, is a general methodology to improve the performance of any weak classifiers better than random guessing. Combining some of the features of both categories of classifiers, S-AdaBoost [Liu and Loe, 2003a] classifier will be introduced and discussed in detail in the following sections and chapters.

Two main classes of the dynamic ensemble classifiers $\hat{\mathbf{E}}$ are the ME (Mixture of Experts) classifier and the HME (Hierarchical Mixture of Experts) classifier. Input patterns $\mathbf{X_i}$, together with the outputs of the individual classifiers $\hat{\mathbf{w}}_i$s, jointly act as the inputs to the final combiner, which generates the final classification result output (as shown in Figure 2.2). In the ME classifier, all of the outputs from the individual classifiers $\hat{\mathbf{w}}_i$s are non-linearly combined (usually the outputs from the individual classifiers are *softmaxed* [Bridle, 1990] before being combined) by one gating network; whereas in HME classifier, outputs from the individual classifiers $\hat{\mathbf{w}}_i$s are non-linearly combined by several hierarchical gating networks before being combined by the final Combiner $\hat{\mathbf{C}}$. Involving the input patterns $\mathbf{X_i}$s of the individual component classifiers to the Combiner $\hat{\mathbf{C}}$ greatly increases the complexity of the algorithm and chance to overfit the input patterns if there are not enough training data available.

It has been reported [Dietterich, 1997] that the ensemble classifier $\hat{\mathbf{E}}$ can often achieve more accurate classification results on benchmark datasets than the individual base classifiers $\hat{\mathbf{w}}_i$ that make it up. It is this discovery that leads to the active research in this direction.

The training of the ensemble classifier $\hat{\mathbf{E}}$ generally begins with the training of a set of individual component classifiers $\hat{\mathbf{w}}_i$ (sometimes they are called the weak learners or the base learners in the Boosting domain or called the hypothesis experts in the Committee Machine domain), followed by the aggregation (or "combination") of the Combiner $\hat{\mathbf{C}}$ to integrate the classification results of these individual component classifiers $\hat{\mathbf{w}}_i$s. The common methodology of choosing the most suitable individual component classifiers $\hat{\mathbf{w}}_i$ is based on the principle of generating more diversity among the individual component classifiers $\hat{\mathbf{w}}_i$s. This is due to the research result that [Hansen & Salamon, 1990] a necessary and sufficient condition for an ensemble classifier $\hat{\mathbf{E}}$ to be more accurate than any of the individual component classifiers $\hat{\mathbf{w}}_i$ that makes the ensemble classifier $\hat{\mathbf{E}}$ up is that the individual component classifiers $\hat{\mathbf{w}}_i$s are accurate and diverse. The definition of the individual component classifiers $\hat{\mathbf{w}}_i$s being "accurate" in this content is that every individual component classifier's performance is better than random guessing; and the definition of the individual component classifiers $\hat{\mathbf{w}}_i$s being "diverse" is that the individual component classifiers $\hat{\mathbf{w}}_i$s can make different kinds of errors on the same new input patterns. It is evident that it is relative easier to construct an "accurate" classifier than a "diverse" classifier.

Approaches from different viewpoints have been proposed to construct the individual component classifier $\hat{\mathbf{w}}_i$s to create diversities. Starting from Bayesian voting based approach [Neil, 1993], which initially proposed to enumerate the individual component classifiers in an ensemble machine with very limited success, four main categories of approaches have since been developed: approaches based on the manipulation of the input training patterns $\mathbf{x}_i$s; approaches based on the manipulation of the input feature sets of input training patterns $\mathbf{x}_i$s; approaches based on the manipulation of the output patterns $\mathbf{Y}$**;** and approaches based on the methodologies injecting the randomness directly to the algorithm $\hat{\mathbf{w}}_i$ itself to create diversity.

Approaches based on the manipulation of the input training patterns $\mathbf{x}_i$s works well for the ensemble classifiers whose component classifiers $\hat{\mathbf{w}}_i$s are unstable, which means that the minor change of the training input pattern $\mathbf{x}_i$ results in the major variation of the classification output $\mathbf{Y}$. Typical examples of the unstable base classification algorithms $\hat{\mathbf{w}}_i$s are neural network algorithm [Schwenk H. and Bengio Y., 1997; Schwenk H. and Bengio Y., 2000] and decision-tree algorithm. Among all the algorithms, random replacement Bagging (which stands for "bootstrap aggregation") [Breiman, 1996], leave-one-out cross-validation committee machine [Parmanto, Munro, Dayle, 1996], and the AdaBoost algorithm are three representative algorithms belonging to the manipulation of input training patterns $\mathbf{x}_i$s category. The second category of approaches based on the manipulation of the input features only works well when the numbers of the input features are highly redundant [Tumer and Ghost, 1996].

Two typical examples, ECOC (Error-Correcting Output Codes) and the AdaBoost.OC (AdaBoost.OC is the combination of ECOC and the AdaBoost algorithm) [Schapire, 1997] fall into the third category manipulating the output classification result **Y**. The last category works by injecting randomness directly to the individual component classification algorithms $\hat{\mathbf{w}}_i$s. Neural Network [Kolen & Pollack, 1991], C4.5 [Kwok and Carter, 1990; Dietterich 2000], and FOIL [Ali and Pazzani, 1996] can be used as the algorithm receiving the random noise injection to generate the required diversity.

Based on the different combination mechanisms used, the Combiner $\hat{\mathbf{C}}$ can be categorized into: combiners based on the combination by voting mechanism (used by the Bagging, the ECOC, and the AdaBoost algorithms) and combiners based on the combination by confidence value (techniques used including stacking [Breiman 1996; Lee and Srihari, 1995; Wolpert, 1992], serial combination [Madhvanath and Govindaraju, 1995], and weighted algebraic average [Jacob, 1995; Tax et al., 1997]).

In the past few years, many ensemble algorithms have been proposed. Among them, some of the leading algorithms are Bagging [Breiman, 1996], Boosting and AdaBoost [Freund & Schapire, 1999], ECOC (Error-Correcting Output Codes) [Dietterich & Bakiri, 1995]. Among those approaches based on these leading algorithms, the AdaBoost algorithm-based approaches often outperform the approaches based on other algorithms [Dietterich, 2002]. The AdaBoost based ensemble classifiers are gaining more and more popularity due to their simplicity and effectiveness in solving problems.

2.2 Face Detection and Face Identification in a Complex Environment

Face Detection [Yang, Kriegman, and Ahuja, 2002; Viola P. and Jones M., 2001] and Face Identification [Zhao, Chellappa, Rosenfeld, and Phillips, 2000a; He. X, Yan S., Hu and Zhang H.J., 2003] are two active research topics under the regime of pattern recognition. Face detection can be considered as the first step towards a face identification or recognition system, but this first step is in no way less challenging than the face identification system itself.

In statistical learning, to estimate a classification decision boundary using a finite number of training patterns implies that any estimate is always inaccurate (biased). For a complex pattern classification problem (like face detection or face identification), it is becoming more and more difficult to collect enough and good training patterns. Non-perfect training samples will increase the complexity of the input space and results in a problem commonly known as “curse of dimensionality”. In the absence of any assumption or empirical knowledge about the nature of the function, the learning problem is often ill-posed. In statistical learning theory, the “divide and conquer” strategy is a means to solve this “curse of dimensionality”.

Face pattern detection [Li, Zhu, Zhang, Blake, Zhang and Shum, 2002; Pentland, 2000a; Pentland 2000b; Pentland and Choudhury, 2000; Viola P. and Jones M., 2001] can be regarded as a two-class pattern classification (“face” v.s “non-face”) task. Face detection is to determine and locate all face occurrences in any given image. A face detection system extracts potential face regions from the background. A complex environment including differences in scale, location, orientation, pose, expression, occlusion and illumination associated with the face pattern detection often makes the face detection task challenging. Feature-based approaches and statistical approaches are two major types of algorithms used to detect faces. Feature-based approaches are further divided into knowledge-based approaches [Kanade, 1973; Kotropoulos and Pitas, 1997; Pigeon and Vandendrope 1997; Yang and Huang, 1994], feature invariant approaches [Kjeldsen and Kender, 1996; Leung, Burl and Perona 1995; McKenna, Gong and Raja, 1998; Yang and Waibel, 1996; Yow and Cipolla, 1997] and template matching approaches [Venkatraman and Govindaraju, 1995; Sinha, 1995; Lanitis, Taylor and Cootes, 1995; Govindaraju, Srihari and Sher, 1990; Craw, Tock and Bennett, 1992]. In the first category, human face features and their relationships are coded in the database or templates, and the correlations between the new image and the feature sets are calculated to determine whether the new image is a face or not. The second statistical approach category takes a holistic approach to the face detection task; it is also referred as appearance-based method in some literature. In contrast to comparing the new input with the fixed stored features as done in the first category, approaches in this category make use of statistical learning and machine learning techniques to establish a model of a face through learning the face knowledge from a known set of training patterns. The learned implicit knowledge is then embedded in the distribution model or the discriminant functions (including the decision boundaries, the separating hyper-planes or the threshold functions) that are later used to detect faces from new input images. The popular approaches, which utilize the PCA [Turk and Pentland, 1991], Support Vector Machine [Osuna, Freund and Girosi F, 1997], Gaussian distribution [Sung and Poggio, 1998], Naive Bayes statistics [Schneiderman and Kanade, 1998], Hidden Markov Model [Rajagopalan, Kumar, Karlekar, Manivasakan, Patil, Desai, Poonacha and Chaudhuri, 1998)], Entropy theory [Lew, 1996] and neural networks [Rowley, Baluja and Kanade, 1998] fall into this category.

Various error rates are used to describe the effectiveness of face detection algorithms. Two commonly used error rates are: the *false negative rate*, which measures the error rate of “faces” being wrongly classified as “non-faces”; and *false positive rate*, which measures the error rate of “non-faces” being wrongly classified as “faces”. A fair measure should take both the above rates into consideration, as reducing one rate might result in increasing the other rate. In this thesis, “the detection error rate” is used to measure the effectives of an algorithm, which is defined as the number of all the wrongly classified cases (including both the number of cases of “faces” wrongly classified as “non-faces” and the number of cases of “non-faces” wrongly classified as “faces”) divided by the number of all cases.

Another issue is the definition of “face detection”. Some definitions are based on the existence of certain features and some definitions follow the judgment of human beings. But it is understood that human beings are sometimes even ambiguous among ourselves about whether a particular cut-out of an image is a face or not. All the above makes the face detection task very challenging. In the experiment, an international airport (as shown in Figure 2.3) is used as the testing complex environment, where thousands of people pass by everyday. The training and testing image patterns are taken by the CCD Cameras installed there.

Figure 2.3 Typical scenarios in the complex airport environment

The face recognition or face identification system [Zhao, Chellappa, Rosenfeld and Phillips, 2000] is a non-intrusive biometric system being able to conduct the identification or the recognition of a number of candidates from a crowd. The facial recognition or the identification system can be used for criminal or idendity recognition purposes.

Similar to the face detection task, there are also two main methodologies behind all the approaches: feature-based method and statistical method. Feature-based face identification systems are built on the analysis of the potential human face sub-images of an input image for the purpose of identification. By measuring the existence of certain facial characteristics (such as the distance between the eyes, the length of the nose, the angle of the jaw), the feature-based face identification systems create a unique file called a "template file". Using templates stored in the template file, the systems can compare the new input image with the stored face templates and produce a score that measures how similar the new image is to the stored face images. The scores obtained are used to make judgment on deciding whether the new input is a face image. Another more popular methodology is based on the statistical property of the image, which attracts active research attention. Similar to the former approach, a segmented potential face image is fed into the statistical identification module, which reports back the determined identity if the identification module finds a match from a database of the known candidates. The statistical identification module is trained by the known input patterns; the feature's known characteristics and other unknown hidden characteristics are coded in the distributed mechanism embedded in the module itself. Enhanced face identification is also studied with the aid of the known information like human race, gender, and speech characteristics to assist the identification. We'll not touch on the enhanced face identification methods in this thesis.

In a complex environment, the challenge to a face recognition/identification system comes from the variances in image background, occlusion, and hairstyle; besides the two well-known difficulties, which are the variation of the background illumination and the difference of the poses.

To handle the complex environment, various methodologies have been proposed. Based on the behavior of certain characteristics of noise, some heuristic methods (such as discarding the smallest principle components in the Eigen-face approach [Belhumeur, Hespanha and Kriegman, 1997; Turk and Pentland, 1991]) achieve good results in reducing the influence of the background illumination, the symmetry feature of the face pattern is also used in some approaches (such as [Zhao, 1999]) to reduce the influence of the noise in the complex environment. These noise pattern based approaches apparently are very dependent on the environment itself and might not function well in a simple environment.

Many approaches have been proposed to tackle face recognition in the complex environment dominated by the illumination variation. Based on the statistical knowledge that the difference of the same face in a different environment is smaller than the difference between two different faces, some image comparison based approaches (such as [Jacobs, Belhumeur and Basri, 1998]) are developed to tackle the complex environment, but these approaches are not capable of handling the complex environment effectively by themselves. Class-based approaches (such as [Belhumeur and Kriegman, 1997]) assume that the face images are of the Lambertain surface without shadowing; three faces under different lighting conditions are obtained to construct a 3D model, which is invariant to lighting and other kinds of noises. Model-based approaches (such as [Atick, Griffin and Redlich, 1996]) use PCA analysis and ICA analysis to transfer the Shape-From-Shading problem into a parametric problem and use many viewpoint samples to construct a model good at handling complex environments.

Developing the face recognition methods in the complex environment that are able to handle multiple types of noise is a current hot topic of research nowadays. The neural network based EBGM (Elastic Bunch Graph Matching) approach [Wiskott, Fellous and Malsburg, 1997], the statistical subspace LDA (Linear/Fisher Discriminant Analysis) approach [Zhao, Chellappa and Krishnaswamy, 2000], and the Probabilistic PCA (Principle Component Analysis) approach [Moghaddam, 2002] are three of the most effective face recognition/identification methods. The EBGM approach defines a planar surface patch in each key landmark location, and studies the transformation of the rotation of the face and pose variation of the images. The system is good at handling face rotation and pose variation through applying techniques like face localization, landmark detection. By defining a graph matching mechanism, the system achieves good experimental results. However the challenge to the EBGM approach is how to accurately locate the landmark points. Statistical sub-space LDA approach aims to reduce the overfitting phenomenon on a large face database. This approach is more suitable for a database with a large number of classes to be classified; in the same time, the database is also under the restriction that only a small number of training patterns belong to a particular class. Utilizing PCA (Principal Component Analysis), the high dimension face images are projected to the face subspace with a lower dimension in the Statistical sub-space LDA approach. The LDA (Linear Discriminant Analysis) process is conducted upon the PCA vectors in the sub-space. Something unique in the statistical sub-space LDA approach is that the dimension of the face sub-space is fixed regardless of the dimension of the face images, which are normally very big. The face sub-space dimension is decided by the number of the Eigenvectors. Utilizing Kernel PCA techniques, the probabilistic PCA applies a non-linear mapping to the input space and converts the non-linear face identification task to a linear PCA task in the larger dimensional mapped space. The advantage of the Probabilistic PCA approach over the neural network approach is that it reduces the overfitting and does not require optimization. Neither the prior knowledge of the network structure nor the size of the dimension is needed in this appraoch. Typical kernel functions used in the approach are Gaussian functions, Polynomials and Sigmoid functions (Yang Ming-Hsuan, Kriegman David, and Ahuja Narendra. 2002). Another emerging technique, which is called Laplacianface (He. X, Yan S., Hu and Zhang H.J., 2003), takes into account the face manifold structure to recognize faces.

In this thesis, we introduce the S-AdaBoost algorithm. The S-AdaBoost algorithm's effectiveness is demonstrated by the experimental results conducted on some benchmark databases through comparing with other leading outlier handling approaches. To further demonstrate the effectives of the S-AdaBoost algorithm in the real world environment, two application systems, FDAO and FISA are developed.

Chapter Three

Ensemble Boosting

3.1 Ensemble Boosting

Ensemble Boosting (or Boosting) classifier **B** [Schapire, 1990] is a kind of learning classifier Ê defined as the ensemble that combines some weak learners $\mathbf{h}_i$s (also called the weak hypotheses, base classifiers, individual component classifiers, or component classifiers in the boosting theory) to improve the performance of the weak learners. In the process, new weak learners in the ensemble are generated and conditioned on the performance of the previously built weak learners.

There are three main types of boosting classifiers **B**, which are boosting by filtering classifiers (such as [Schapire, 1990]), boosting by sub-sampling classifiers (such as [Freund and Schapire, 1996a]) and boosting by re-weighting classifiers (such as [Freund Y., 1995]). The boosting by filtering classifiers use different weak classifiers $\mathbf{h}_i$s to filter the training input patterns $\mathbf{x}_i$s; the training input patterns $\mathbf{x}_i$s will either be learnt or discarded during filtering. The filtering approach is simple but often requires a large (in theory, infinite) number of training patterns from the training set **X**. Collecting such a large number of training patterns is often impossible in the real world. Compared with the large set of training patterns required in the boosting by filtering classifiers, only a limited set of training patterns $\mathbf{x}_i$s are required in the boosting by sub-sampling classifiers. The training patterns $\mathbf{x}_i$s are re-used and resampled according to certain distribution patterns in the boosting by sub-sampling based approaches. The boosting by re-weighting classifiers also make use of a limited set of training patterns (similar to the boosting by sub-sampling approaches), the difference between these two types of classifiers is that the boosting by re-weighting classifiers receive weighted training patterns $\mathbf{x}_i$s rather than the sampled training patterns $\mathbf{x}_i$s used in the boosting by sub-sample classifiers.

Boosting was originally developed from the Probably Approximately Correct (PAC) theory [Valiant, 1984]. It is proven [Kearns M., and Valiant L.G., 1994] that the Boosting classifier $\mathbf{B}$ can achieve arbitrary good classification results from slightly better than random guessing weak learners $\mathbf{h}_i$s through the boosting process, provided that there is enough training data available. After the first polynomial time boosting classifier $\mathbf{B}$ [Schapire, 1990] was proposed, the first Boosting-based application system [Drucker, H., Schapire, R., and Simard, P., 1993] tackling the real world OCR task was built using a neural network as the base weak learners $\mathbf{h}_i$. In the following paragraphs, it will be explained why the boosting algorithm can boost the performance of the base weak classifier and why a weak classifier is equivalent to a strong classifier in the Boosting framework. The answers to these questions constitute the foundation of the boosting theory.

The PAC learning model is a probabilistic framework for learning and generalization in the binary classification system, and it is closely associated with the supervised learning methods. In the PAC classification learning, the learning machine $\acute{\mathbf{L}}$ tries to conduct classification on the randomly chosen training input patterns with an underlying distribution. The goal of the learning machine $\acute{\mathbf{L}}$ is to be able to classify a problem with an error rate less or equal to an arbitrary small positive number ε, and this property must hold uniformly for all the possible input distributions. As the training input pattern distributions are randomly chosen, the above goal can be achieved with a certain probability, which is defined to be equal to 1 - δ (δ is a small positive number, which is used to measure the unlikelihood of the learning machine **Ĺ** being accurate). The above PAC learning is often called strong learning (As shown in Figure 3.1).

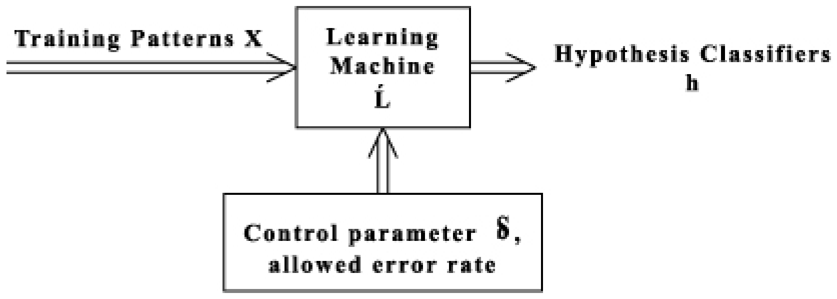

Figure 3.1 PAC Learning model

As the accuracy requirement to the individual weak learners **h**$_i$s in boosting classifier **B** is "slightly better than random guessing", which means that the individual weak learners **h**$_i$s are only required to achieve slightly better than ½ accuracy in the binary classification; the requirement to the base learning algorithm is dramatically relaxed in the boosting algorithms. This kind of learning used in boosting algorithms is called weak learning compared with the PAC strong learning described in the above paragraph.

Schapire [1990] proved constructively that weak learning and strong learning are equivalent. A boosting by filtering classifier **B** with three individual weak learners **h**$_i$s can convert an arbitrary weak learning classifier to a strong learning classifier (one of the conversion processes is shown in Figure 3.2).

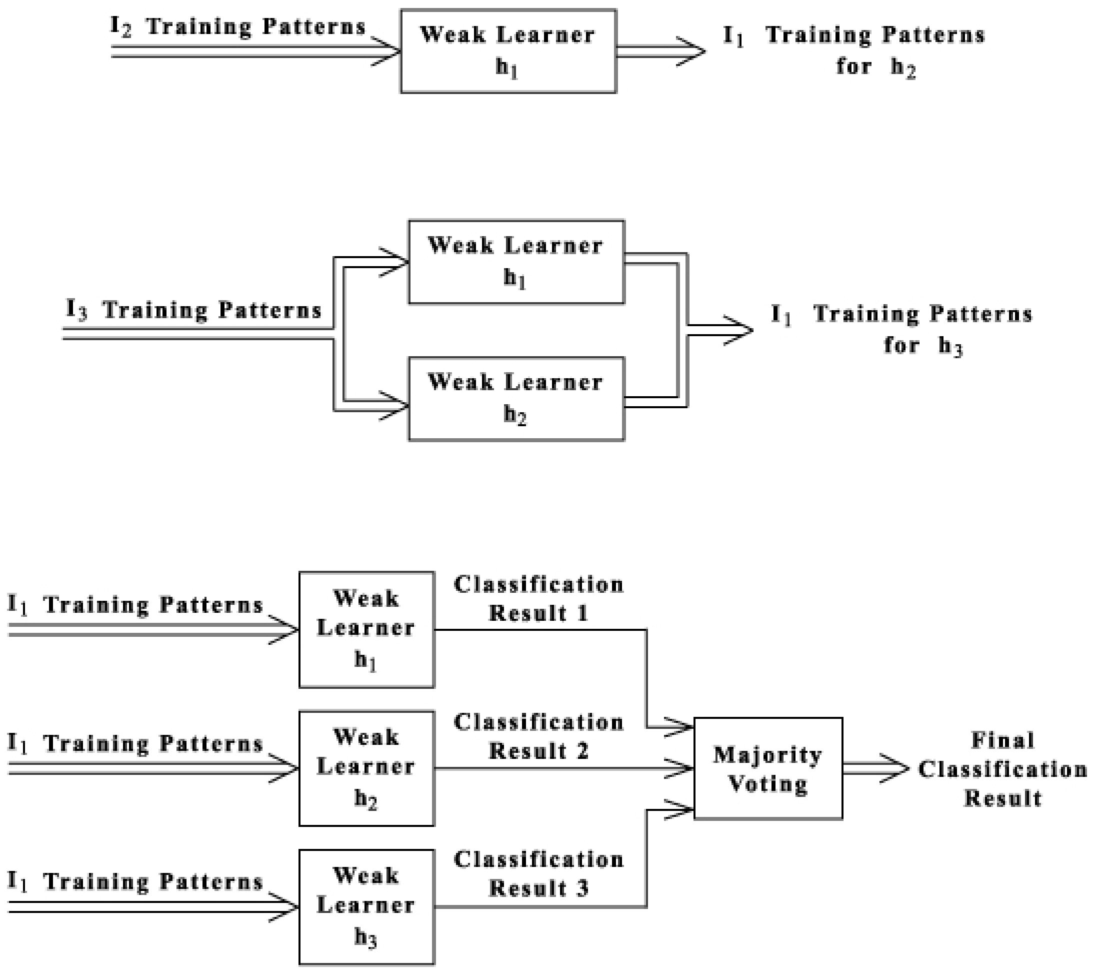

Figure 3.2 Boosting by filtering - a way of converting a weak classifier to a strong one

From Figure 3.2, it is shown that the first step of the boosting by filtering algorithm is to train the individual weak learner $\mathbf{h}_1$using the $\mathbf{I}_1$ training patterns randomly chosen from the input pattern set **X**. The method of obtaining the $\mathbf{I}_1$ training patterns, which will be used to train the weak learner $\mathbf{h}_2$ can be described as:

Initialize the number of the training patterns already obtained for the weak learner $\mathbf{h}_2$ to 0:

i = 0.

Get a function *Random* (), which can generate values 0 and 1 randomly.

Use $\mathbf{y}_1(\mathbf{x})$ to represent the targeted output of the weak learner $\mathbf{h}_1$.

```
Use $\mathbf{X}_2$ to represent the training set needed for training the weak
        learner $\mathbf{h}_2$,
initialize the set $\mathbf{X}_2$ by setting:
$\mathbf{X}_2 = \{\}$;
Use $\mathbf{h}_1(\mathbf{x})$ to represent the actual output of the individual weak
        learner $\mathbf{h}_1$.
LOOP until (i = $\mathbf{I}_1$)
        BEGIN
                IF (*Random* () ≡ 1)
                BEGIN
                LOOP until $\mathbf{h}_1$(new training pattern $\mathbf{x}$) $\neq$ $\mathbf{y}_1(\mathbf{x})$)
                        BEGIN
                        Get a new training pattern x.
                        END
                END
                ELSE
                BEGIN
                LOOP until $\mathbf{h}_1$(new training pattern $\mathbf{x}$) $\equiv$ $\mathbf{y}_1(\mathbf{x})$)
                BEGIN
                Get a new training pattern x.
                END
        END
        i = i + 1;
        Store the current $\mathbf{x}$ training pattern in $\mathbf{X}_2$ by setting:
        $\mathbf{X}_2 = \mathbf{X}_2 + \{\mathbf{x}\}$;
END
OUTPUT $\mathbf{X}_2$
```

The output $\mathbf{X}_2$ set contains the $\mathbf{I}_1$ training patterns used to train the weak learner $\mathbf{h}_2$ in the future.

In this way, all the $\mathbf{I}_1$ training patterns, which are used to train the individual weak learner $\mathbf{h}_2$ are of different distribution from the $\mathbf{I}_1$ training patterns, which have been selected to train the individual weak learner $\mathbf{h}_1$. If this $\mathbf{I}_1$ training patterns, which are used to train individual weak learner $\mathbf{h}_2$, is used to test the individual weak learner $\mathbf{h}_1$, a 0.5 error rate will be obtained.

Similarly, the requirement for getting the $\mathbf{I}_1$ training patterns for the individual weak learner $\mathbf{h}_3$ is that the new $\mathbf{I}_1$ training patterns must be of different distribution comparing with the $\mathbf{I}_1$ training patterns that are used to train the individual weak learner $\mathbf{h}_1$ as well as the $\mathbf{I}_1$ training patterns that are used to train the individual weak learner $\mathbf{h}_2$. The method can be described as:

```
Initialize the number of the training patterns already obtained for the weak
learner $\mathbf{h}_3$ to 0:
    i = 0.
Use $\mathbf{X}_3$ to represent the training set needed for training the weak learner $\mathbf{h}_3$,
initialize the set $\mathbf{X}_3$ by setting:
    $\mathbf{X}_3 = \{\}$;
Use $\mathbf{h}_1(\mathbf{x})$ to represent the actual output of the individual weak learner $\mathbf{h}_1$.
Use $\mathbf{h}_2(\mathbf{x})$ to represent the actual output of the individual weak learner $\mathbf{h}_2$.
    LOOP until (i = $\mathbf{I}_1$)
        BEGIN
        LOOP until $\mathbf{h}_1$(new training pattern $\mathbf{x}$)
                $\neq \mathbf{h}_2$(new training pattern $\mathbf{x}$))
            BEGIN
            Get a new training pattern $\mathbf{x}$.
            END
        i = i + 1;
        Store the current $\mathbf{x}$ training pattern in $\mathbf{X}_3$ by setting:
        $\mathbf{X}_2 = \mathbf{X}_3 + \{\mathbf{x}\}$;
        END
    OUTPUT $\mathbf{X}_3$
```

The output $\mathbf{X}_3$ set contains the $\mathbf{I}_1$ training patterns used to train the weak learner $\mathbf{h}_3$ in the future.

Through this way, all the $\mathbf{I}_1$ training patterns, which are used to train the individual weak learner $\mathbf{h}_3$ are of different distribution from both the $\mathbf{I}_1$ training patterns, which have been selected to train the individual weak learner $\mathbf{h}_1$ and the $\mathbf{I}_1$ training patterns, which have been selected to train the individual weak learner $\mathbf{h}_2$. If these $\mathbf{I}_1$ training patterns, which are used to train individual weak learner $\mathbf{h}_3$, is to test the individual weak learner $\mathbf{h}_1$ and individual weak learner $\mathbf{h}_2$, a 0.5 error rate will be obtained.

In the following discussion:

$\mathbf{I}_2$ is used to denote the number of training patterns in the input space $\mathbf{X}$ needed to generate the $\mathbf{I}_1$ training samples for training the individual weak learner $\mathbf{h}_2$.

$\mathbf{I}_3$ is used to denote the number of training patterns in the input space $\mathbf{X}$ needed to generate the $\mathbf{I}_1$ training samples for training the individual weak learner $\mathbf{h}_3$.

The total number of training patterns needed to train the boosting by filtering classifier $\mathbf{B}$ is:

$$\mathbf{I} = \mathbf{I}_1 + \mathbf{I}_2 + \mathbf{I}_3 \tag{3.1.1}$$

From the above discussion, it is known that this number $\mathbf{I}$ can be very big sometimes. In the statistical learning theory, the VC Dimension (the Vapnik Chervonenkis Dimension) provides some theoretical foundation to estimate the number of $\mathbf{I}$, which is the optimal size of the training set. In the PAC contents and neural network implementation, the following statement is proposed (Blumer, Ehrenfeucht, Haussler and Warmuth, 1989; Anthony and Biggs, 1992; Vidyasagar, 1997):

In the PAC framework, for a neural network implementing any constant learning algorithm with a finite VC Dimension $\ddot{\upsilon}$ ($\ddot{\upsilon}$ is equal to or greater than one), a constant $\mathbf{Ķ}$ exists such that the sufficient size of patterns in the training input space is:

$$\mathbf{I} = \frac{\mathbf{Ķ}}{\varepsilon}\left(\ddot{\upsilon}\log\left(\frac{1}{\varepsilon}\right) + \log\left(\frac{1}{\delta}\right)\right) \qquad (3.1.2)$$

Where,

δ is the confidence parameter used to measure the unlikelihood of the learning machine being accurate.

ε is the error parameter.

Under the boosting by filtering framework, assuming that the error rates of the three individual weak learners are the same, it is proven [Schapire, 1990] that the overall error rate of the boosting by filtering classifier **B** is bounded by

$$\hat{ẹ} = 3\varepsilon^2 - 2\varepsilon^3 \qquad (3.1.3)$$

Where ε stands for the error rate of the individual weak classifier and its values is less than ½.

The bounding error $\hat{ẹ}$ is demonstrated in Figure 3.3, from where it can be found that the new error $\hat{ẹ}$ is smaller than the original error rate ε of the individual weak classifiers. It is also meant that by applying the above boosting process recursively, arbitrary small error rate for the ensemble can be achieved. In this way, it is proven that weak learning is equivalent to strong learning.

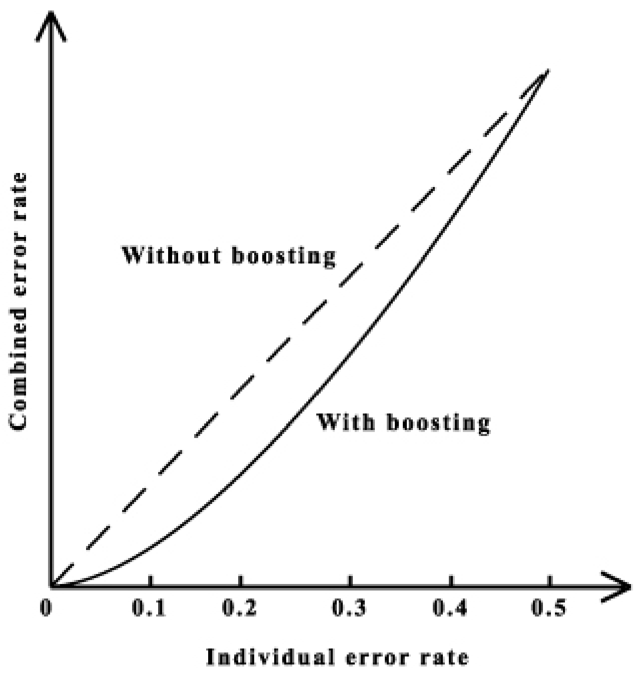

Figure 3.3 Boosting combined error rate bounding

In the following, $\hat{\mathbf{C}}(\mathbf{h}(\mathbf{x}))$ is used to denote the final hypothesis generated by the ensemble boosting classifier **B** on the training patterns, which represents the input feature set; $\mathbf{h}(\mathbf{x})$ represents the hypothesis weak classifier with $\mathbf{x}$ as its input; and $\hat{\mathbf{C}}(\mathbf{h})$ represents the combination function, combining the output of the hypothesis weak classifier **h**. In the classification scenario, the output labels $\mathbf{y_i} \in \mathbf{Y} = \{-1, 1\}$ are used to denote the targeted output labels for binary classification (when the output is the scalar value, output labels $\mathbf{d_i} \in \mathbf{D}$ are sometimes used to denote the targeted output labels instead of using $\mathbf{y_i}$). The objective of the boosting machine **B** is to minimize the error rate over all **N** patterns in the test set:

$$\frac{1}{N} \sum_{i=1}^{N} \mathrm{Abs}[y_i - \hat{\mathbf{C}}\ (\mathbf{h}_j(xi))] \qquad (3.1.4)$$

Where Abs [**s**] denotes the absolute value of **s**.

Different boosting classification algorithms can be implemented by choosing different determinations of the Combiner $\hat{\mathbf{C}}$, individual hypothesis classifier $\mathbf{h_j}$ and input pattern selection methods for $\mathbf{h_j}$. This thesis focuses the discussion on building boosting classifiers through supervised learning. Boosting principle used for regression [Duffy and Helmbold, 2000; Ratsch, Demiriz and Bennett, 2002] and unsupervised learning [Allwein, Schapire and Singer, 2000; Ratsch, Smola and Mika, 2003] will not be discussed.

3.2 AdaBoost (Adaptive Boosting)

Even though the original Boosting by filtering algorithm [Schapire 1990] is of little practical use due to its stringent demand for too large a training pattern set, it had led to the evolution of a series of boosting algorithms [Freund and Schapire 1996a; Friedman J., Hastie T. and Tibshirani R., 1998; Freund, Y., 1999; Domingo C. and Watanabe O., 2000; Rätsch G, Onoda T. and Müller K. R., 2001]. Among them, the AdaBoost algorithm (Adaptive Boosting) [Freund and Schapire 1996a] is one of the simplest yet most effective boosting algorithms. The AdaBoost algorithm gains its popularity in practice due to its being simple, effective and adaptive, which means that the update to its distribution weights is a function of the weighted error rates of the individual weak classifier (hypothesis) $\mathbf{h_j}$. While the theory foundation of the AdaBoost algorithm is still being improved, the practical effectiveness of the AdaBoost algorithm has been demonstrated by many researchers.

To overcome the limitation of requiring a large amount of training data in Boosting by filtering approach, the AdaBoost algorithm reuses the samples in every new round of iteration. With a weak learning model $\hat{\mathbf{W}}$, the AdaBoost algorithm targets to find a classifier $\mathbf{h}_{\mathbf{final}}$ with a lower error rate than the error rates of the original individual weak classifiers $\mathbf{h}_{\mathbf{j}}$s.

The pseudo code of the AdaBoost algorithm for a two-class classification can be described as follows:

```
Given: Weak learning algorithm $\hat{\mathbf{W}}$;
        Training patterns: $\mathbf{P} = \{\mathbf{p}_i = (\mathbf{x}_i, \mathbf{y}_i)\}$ for i = 1 to M
                Where M stands for the number of the training patterns;
                $\mathbf{x}_i \in \mathbf{X}$ stands for the input patterns;  and
                $\mathbf{y}_i \in \mathbf{Y} = \{-1,1\}$ stands for the targeted output;
        Number of iteration T;
L1: Initialize distribution $D$:
        Set $D_1(i) = \frac{1}{M}$, for all i = 1 to M;
        Set iteration count t = 1;
        Set initial error rate $\in_1 = 0$.
L2: Iterate while $(\in_t < 0.5)$ and $(t \le T)$
        -  Call $\hat{\mathbf{W}}$ algorithm with distribution $D_i$
                Obtain the hypothesis $\mathbf{h}_t : X \to Y$
        -  Calculate the weighted error rate:
                $\in_t = \sum_{i:h_t(x_i) \neq y_i} D_t(i)$
        -  Set $\beta_t = \frac{\in_t}{(1 - \in_t)}$
        -  Update the new distribution D for i= 1 to M:
                $D_{t+1}(i) = \frac{D_t(i)\beta_t^{Sign(h_t(x_i) == y_i)}}{z_t}$;
        Where $z_t$ is a normalization factor chosen such that the new distribution $D_{t+1}$
        is a normalized distribution, where $\text{Sign}(x) = \begin{cases} 1 & \text{if } X > 0 \\ 0 & \text{if } X \le 0 \end{cases}$.
    -   $t = t + 1$;
L3:   Calculate the final hypothesis classifier
```

$$\mathbf{h}_{\text{final}}(x)) = \underset{y \in Y}{\text{ArgMax}} \sum_{t=1,\, h_t(x)=y}^{T} \text{Log}\left(\frac{1}{\beta_t}\right)$$

From the above pseudo code, it is shown that on iteration t, the AdaBoost algorithm learns the training input patterns **P** with the distribution D_t, a new hypothesis $\mathbf{h}_t$ is also returned. New error rates are calculated with respect to the distribution D_t. The process will continue until the error rate is no more than 0.5 or the number of iteration reaches the specified number T. The final hypothesis classifier $\mathbf{h}_{\text{final}}$ is calculated by combining all the classification results of the past individual hypothesis classifiers $\mathbf{h}_1$**,** $\mathbf{h}_2$**,** to $\mathbf{h}_T$. For distribution weights, starting from the uniform distribution D_1, the new distribution D_{t+1} is calculated based on the previous distribution D_t and the classification results of the previous individual hypothesis classifier $\mathbf{h}_t$. If the individual hypothesis classifier $\mathbf{h}_t$ classifies the new input pattern correctly, the new distribution weight is adjusted with a factor less than one; otherwise the new distribution weight remains unchanged before normalization. In this way, the distribution weights of the training patterns are adjusted each round so that the individual weak hypothesis classifiers focus more and more on the *difficult* training input patterns. The final hypothesis classifier $\mathbf{h}_{\text{final}}$ is calculated based on the weighted vote from all the previous individual hypothesis classifiers $\mathbf{h}_t$s. Greater voting weights are assigned to those $\mathbf{h}_t$s with lower error rates.

To summarize, in classification scenario, the Adaptive Boosting (AdaBoost) algorithm runs a given weak learning algorithm $\hat{\mathbf{W}}$ repeatedly for certain rounds, the distribution weights of the training patterns are adjusted each round so that the individual hypothesis classifiers $\mathbf{h_t}$s focus more and more on the *difficult* training patterns. In this way, the performance of the ensemble is boosted.

Assuming the error rates $\boldsymbol{\varepsilon}$ generated by the individual hypothesis classifiers (as defined in the AdaBoost pseudo code) satisfy $\boldsymbol{\varepsilon_t} \leq ½$ (t = 1 to T), it can be proven [Freund and Schapire 1996a] that the upper bound of the error rate $\boldsymbol{\varepsilon_{final}}$ of $\mathbf{h_{final}}$ is:

$$\text{Exp}\left(-2 \sum_{t=1}^{T} \left(\frac{1}{2} - \varepsilon_t \right)^2 \right) \tag{3.2.1}$$

This indicates that if $\mathbf{h_t}$s, which are constructed on the weak learning algorithm $\hat{\mathbf{W}}$, have error rates only slightly better than ½, the error rate $\boldsymbol{\varepsilon_{final}}$ of the final hypothesis classifier $\mathbf{h_{final}}$ drops to zero exponentially fast.

However, the above upper theoretical bound could not provide much practical guideline. Experiments conducted [Freund and Schapire 1996a] have indicated that the theoretical bound on the training error is too weak and the generalization error tends to be much better that what the theory would suggest; the AdaBoost related researches have since been focused on the experimental results [Haykin S. p362, 1998].

The effectiveness of the AdaBoost algorithm can also be explained from the Gaming theory [Breiman, 1997], the Statistical Learning theory and the VC theory [Schapire, Freund, Bartlett and Lee, 1998] as well as from the Information theory's entropy analysis [Kivinen and Warmuth, 1999] other than the PAC theory.

Breiman [1996] noted that one very unique feature of the AdaBoost algorithm is that the error rate for testing in the AdaBoost algorithm continues to drop after the error rate for training has decreased to nearly zero during the training process (as shown in Figure 3.4). This phenomenon has also been observed in the Boosting by filtering algorithm [Druker, Cortes, Jackel and LeCun, 1994]. People have been puzzled with this unique feature, as this phenomenon is seemingly controversial to the common belief of the generalization performance of a learning machine. According to the well known *Occam's razor*, (stated by the 14th century logician and Franciscan friar, William of Occam) a learning machine should be as simple as possible to achieve good generalization result ("Entities should not be multiplied unnecessarily"), people normally feel that in a learning machine, along the decreasing of the training error rate, the testing error rate will drop to a minimum before climbing up due to overfitting (as shown in Figure 3.5). The phenomenon shown in Figure 3.4 shows that the generalization error decreases further even when the system has been over-trained in the AdaBoost algorithm. Recently, there are some breakthroughs in theory in explaining the above unique feature, the theoretical explanation [Schapire, Freund and Bartlett, 1997] based on the "margin" and "confidence of the classifier" concepts show that the above phenomenon is related to the distribution of the margins of the training input patterns with respect to the classification error.

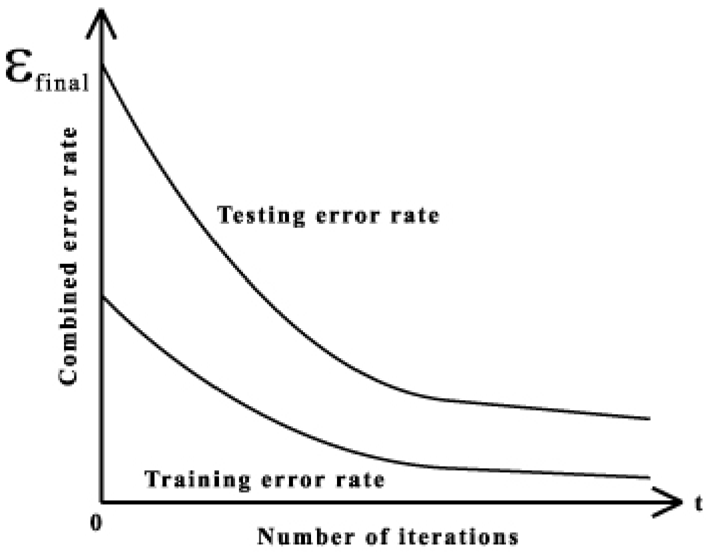

Figure 3.4 The AdaBoost machine's performance

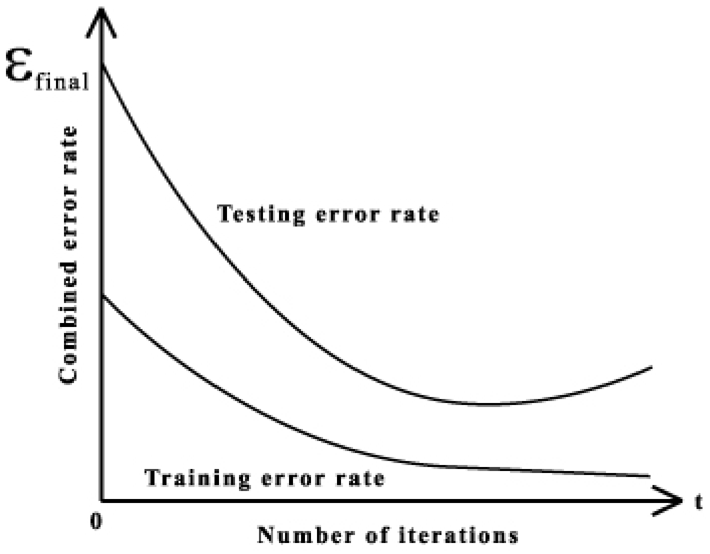

Figure 3.5 Normal learning machine's performance

Following the initial success of the original binary AdaBoost algorithm, researchers have tried various means to improve the performance of the algorithm and extend the application domains of the original AdaBoost algorithm. Nowadays, the AdaBoost algorithm has been successfully extended from the binary classification to multi-class classification [Freund and Schapire, 1997; Schapire, 1997; Schapire and Singer, 1998]. Adjusting the parameters in the original binary AdaBoost algorithm, different error function-based approaches including the LogitBoost's binomial loglikelihood approach [Friedman, Hastie and Tibshirani, 2000], the Gentle AdaBoost's Newton step approach [Friedman, Hastie and Tibshirani, 2000] have also developed and experimented with some success. The selection of the base weak classifier has also attracted much attention from the researchers.

The selection of the base weak classifier plays a very important role in determining the performance of the AdaBoost algorithm, a good base weak classifier should be simple yet not too weak. The AdaBoost algorithm may encounter early stopping (error rate <1/2 for the binary classification case) for a too weak base classifier; and a too complex base classifier may lead to overfitting and the demanding of too big a training pattern set. Researchers experiment with different base weak classifiers (or called base classifiers or weak learners) such as CART (the Classification And Regression Tree) [Breiman, Friedman, Olshen and Stone, 1984], C4.5 [Quinlan, 1992], Neural Networks [Haykin, 1998; Bishop, 1995], etc. Among them, the Decision Trees and Stump Decision Trees are becoming two popular approaches due to their easy implementation and effectiveness [Hastie, Tibshirani and Friedma, 2001; Quinlan, 1992]. The Decision Tree based approaches are based on the nested recursive division of the input space while the Stump Decision Tree based approaches are based on a single layer tree structure. Approaches based on both pure Decision Tree [Schapire and Singer, 1998] and Decision Tree with logistic regression [Friedman, 1999] have also been explored and have achieved good experimental results.

The Neural Network based approaches [Haykin, 1998; Bishop, 1995] have been studied extensively as well. As Neural Network is a relatively strong classifier compared with other weak classifiers mentioned above, theoretically, using Neural

Network as base weak classifier might lead to overfitting of the system. However, studies show [Schwenk and Bengio, 2000] that the AdaBoost algorithm can significantly improve the performance of the Neural Network based weak classifiers in the real world OCR (Optical Character Recognition) application; this is further demonstrated in the AdaBoost algorithm-based S-AdaBoost algorithm's experimental results in facial detection and facial identification in the complex airport environment as well as some benchmark databases [Liu and Loe, 2003a; Liu and Loe, 2003b; Liu, Loe, Zhang 2003c]. Positive results making use of different kinds of weak base classifiers in real world applications will be demonstrated in greater detail in the coming sections and chapters.

3.3 Outliers and Boosting

The AdaBoost algorithm is famous for being effective, adaptive and simple. As demonstrated in the previous section, overfitting seldom happens in the AdaBoost algorithm-based system in low noise applications. However, the AdaBoost algorithm's weakness in handling outliers in the complex environment are also pointed out and discussed [Dietterich and Kong, 1995; Quinlan 1996; Jiang, 2001; Dietterich 2000; Grove and Schuurmans 1998; Rätsch, 1998]. Some research has been conducted [Jiang, 2001; Wyner, Kriege and Long, 2001; Friedman, Hastie and Tibshirani, 2000; Freund, 1999; Domingo and Watanabe, 2000; Rätsch, Onoda, and Müller, 2001] from both theoretical and practical aspects. Some AdaBoost based methods such as the LogitBoost algorithm and Gentle AdaBoost algorithm [Friedman, Hastie and Tibshirani, 1998], BrownBoost algorithm [Freund, 1999] (which is an adaptive version of the Boost-by-Majority algorithm [Freund, 1995]), the MadaBoost algorithm [Domingo and Watanabe, 2000], SmoothBoost algorithm [Servedio, 2001], DOOM

(Direct Optimization of Margins) algorithm [Mason, Bartlett and Baxter, 1998], Regularized AdaBoost algorithm and the LP/QP (linear programming / quadratic programming) AdaBoost algorithm [Rätsch, Onoda and Müller, 2001] and other algorithms are further developed to address and improve the AdaBoost algorithm's capability of handling outliers.

The above classifiers can be categorized into two classes. The first class of classifiers focus more on improving the overall margin distribution of the classifiers, typical examples include the DOOM algorithm, the LP AdaBoost algorithm and the regularized LP AdaBoost algorithm; the second class of classifiers focus more on limiting the influence of the outliers. Typical examples include the Regularized AdaBoost algorithm, the BrownBoost algorithm and the SmoothBoost algorithm; the S-AdaBoost algorithm [Liu and Loe, 2003a; Liu and Loe, 2003b; Liu, Loe and Zhang, 2003c] also falls into this category.

The first category of classifiers are built upon the understanding that the success of the AdaBoost algorithm is due to its capability to boost weak classifiers and produce the final hypothesis classifier with a large margin. In an ideal environment without many outliers, the generalization error of the AdaBoost algorithm decreases with the increment of the size of the margin. However, experiments [Quinlan, 1996; Breiman, 1999; Rätsch, Onoda and Müller, 2001] conducted in the complex environment with many outliers show the discovery that the generalization error does increase with the increment of the size of the margin. It is found that besides the size of the margin, this category of classifiers focus on the good distribution of the margin as well.

The DOOM (Direct Optimization Of Margins) [Mason, Bartlett and Baxter, 1998; Mason 1999; Mason, Baxter, Bartlett and Frean, 2000] algorithm makes use of the different set of loss functions to replace the exponential loss function used by the AdaBoost algorithm in order to achieve a good distribution of margins. A piece-wise linear loss function has been implemented in the DOOM algorithm by some researchers, who find that in theory, optimization of the margin loss function to the global minimum is hardly reachable; but good heuristic improvements [Mason, Bartlett and Baxter, 1998; Mason, 1999; Mason, Baxter, Bartlett and Frean, 2000] can still lead to the delivery of some good results in the complex environment. The drawback of this approach is that a regularization parameter is to be handpicked in the algorithm, which makes the algorithm non-adaptive.

Another two popular approaches belonging to the first category of classifiers is the Linear Programming AdaBoost approach [Freund and Schapire, 1996b; Breiman 1997] and the Regularized Linear Programming AdaBoost approach. The latter one is also called the v-Arc approach [Rätsch, Onoda and Müller, 2001]. The v-Arc approach uses the soft-margin to reformulate the linear programming into a non-linear programming with the maximum hard margin; then another heuristic method is used to maximize the new margin. In the experiment conducted, the effectiveness of the v-Arc approach is demonstrated through comparing its results with those of the AdaBoost algorithm in the noisy environment.

The second category of classifiers try to minimize the influence of the outliers. As introduced before, the weights of those training patterns, which are hard to be classified, are boosted after every round of iteration in the AdaBoost algorithm, and this leads to the increment of the minimum margin for the final combined hypothesis classifier and also results in the decreasing of the training error exponentially. In the AdaBoost algorithm, in an ideal environment without any outliers, a few representative training patterns determine the decision boundary and the size of the margin; in a not so ideal situation with some distortion, the decision boundary and the margin of the AdaBoost algorithm can still be generated without losing the generalization capability through the non-linear transformation, in a situation with some hard-to-classify patterns, the AdaBoost algorithm can still manage to draw a decision boundary with some margin and at the same time overfit the training input patterns or sacrifice the generalization capability; however, in a noisy environment, the noisy patterns will affect both the classifier's accuracy and the generalization capability of the AdaBoost algorithm; the overfitting in this scenario can lead to very poor system performance.

Realizing the above limitations of the AdaBoost algorithm, the Boost-By-Majority (BBM) algorithm [Freund, 1995] was developed. The BBM algorithm aims to minimize the training error within a pre-assigned number of iterations. As the algorithm runs towards its pre-determined number of iterations, the chance of the correct classification of the training patterns with large negative margins becomes slim. The BBM algorithm overcomes this by simply giving up learning from those training patterns with large negative margin and concentrates its effort on those patterns with small negative margins. One of the drawbacks of the BBM algorithm is that a pre-specified upper bound must be specified in the algorithm. In this way, the algorithm is not adaptive, which leads to nearly no practical system being implemented based on this algorithm alone.

The LogitBoost algorithm and Gentle AdaBoost algorithm [Friedman, Hastie and Tibshirani, 1998] were proposed to tackle the limitation of the AdaBoost algorithm by slowing down the speed of weight update. Weight updating schemes in the LogitBoost algorithm and the Gentle AdaBoost algorithm make use of the functions that change not as exponentially fast as that used in the AdaBoost algorithm. In this way, it is hoped that the outliers cannot affect the generalization performance of the algorithms as much as they do in the AdaBoost algorithm. Even though the algorithms slow down the weight increment speed pertaining to the outliers, they still suffer from the overfitting problem when the number of iterations is large.

Trying to overcome the limitation of the non-adaptive nature of the BBM algorithm, the BrownBoost algorithm [Freund, 1999] was proposed as an adaptive version of the BBM algorithm. Taking clue from the Brownian movement in noisy environments, Freund developed the BrownBoost algorithm providing a heuristic method to determine the values of the parameters. The BrownBoost algorithm is an AdaBoost based boosting algorithm with the PAC property. The core idea of the BrownBoost algorithm is to let the error rate approach ½ and observe the possible "movements" of the algorithm. The BrownBoost algorithm performs similar calculations within every round of iteration to that of the AdaBoost algorithm, the difference lies in the ways weights are updated in the BrownBoost algorithm and the AdaBoost algorithm. In the BrownBoost algorithm, a differential equation is to be solved in every round of iteration to calculate the weight changes of every individual hypothesis classifier. Even though the idea looks promising, there is a lack of practical system evidence to support the idea.

The MadaBoost algorithm [Domingo and Watanabe, 2000] was proposed from a slightly different direction. Besides modifying the weights, the algorithm also tries to make itself a boosting algorithm under the Boosting by Filtering framework. Based on the statistical enquiry-learning model, the weights in the MadaBoost algorithm are bounded by their initial probabilities. Domingo and Watanabe claim that the MataBoost algorithm is superior to the AdaBoost algorithm when the number of training patterns is huge (due to its compliance with the Boosting by Filtering framework). By bounding the weights, the weights in the MadaBoost algorithm cannot grow arbitrarily large as happens in the AdaBoost algorithm. The MadaBoost algorithm is slow and needs more experimental results to support the claims.

The SmoothBoost algorithm [Servedio, 2001] was developed based on the idea of smoothing the skew of the pattern distribution to reduce the overfitting. Limitation on the skew is imposed on the algorithm to improve its capability of handling noise in certain conditions. A cut-off mechanism of the pattern distribution weights assigned to the training patterns is used to limit the large negative margins. As there are two parameters (the targeted error rate of the final hypothesis classifier and the guaranteed edge of the hypothesis classifier calculated by the individual weak hypothesis classifier) to be determined beforehand, the algorithm is not fully adaptive. More real world applications are needed to support this promising algorithm.

Among all the approaches, the Regularized AdaBoost algorithm [Rätsch, Onoda and Müller, 2001] is one of the best performers in handling outliers. Realizing the overfitting of the AdaBoost algorithm in the high noise applications and analyzing the boosting from the "soft margin" approach, Rätsch takes clue from the Support Vector Machine [Vapnik 1995; Cortes and Vapnik, 1995] and introduces the *mistrust* to be associated with the training patterns to alleviate the distortion that an outlier can cause to the margin distribution. In the Regularized AdaBoost algorithm, the mislabeled training patterns and the outliers are not given as much attention as the reliable patterns. The mistrust values are calculated based on the weights calculated for those training patterns. The Regularized AdaBoost algorithm (where the gradient descent is done directly with respect to the soft margin) achieves very good experimental results. While achieving good performance, Rätsch's approach needs vast computation resources to get the optimal parameters.

How to find a simple yet effective approach to handle outliers in the Adaptive Boosting algorithm framework is a challenge to the researchers. In the following chapter, the S-AdaBoost algorithm [Liu and Loe, 2003a; Liu and Loe, 2003b; Liu, Loe and Zhang, 2003c] is introduced to improve the outlier handling capability of the AdaBoost Algorithm.

Chapter Four
S-AdaBoost

4.1 Introduction

We propose the S-AdaBoost algorithm [Liu and Loe, 2003a; Liu and Loe, 2003b; Liu, Loe and Zhang 2003c], which is a new extension of the AdaBoost algorithm and is more effective than the conventional AdaBoost algorithm in handling outliers in the pattern detection and the pattern identification applications in the complex real world environment. Utilizing the “divide and conquer” strategy, S-AdaBoost algorithm innovatively uses the AdaBoost’s adaptive distributive weight as a dividing tool to divide the input space into inlier and outlier sub-spaces and use dedicated classifiers to handle the inliers and outliers in the corresponding sub-spaces before non-linearly combining the results of the dedicated classifiers. The S-AdaBoost system is the ensemble of an AdaBoost divider, an AdaBoost classifier, a dedicated classifier for outliers, and a non-linear combiner. Experiments on a number of benchmark databases are conducted to test the effectiveness of the S-AdaBoost algorithm. The experimental results obtained clearly demonstrate the S-AdaBoost algorithm’s effectiveness in those benchmark datasets.

According to the statistical learning theory [Vapnik, V.N, 1995], learning from known finite training samples can be called statistical estimation; and the pattern classification (recognition) can be regarded as the estimation of class decision boundary [Cherkassky V. and Mulier F., 1998].

In a two-class pattern classification scenario, there are always various types of decision boundaries to classify the finite training samples (as shown in Figure 4.1). The challenge is how to choose a decision boundary, which is able to classify the future input well with good generalization capability.

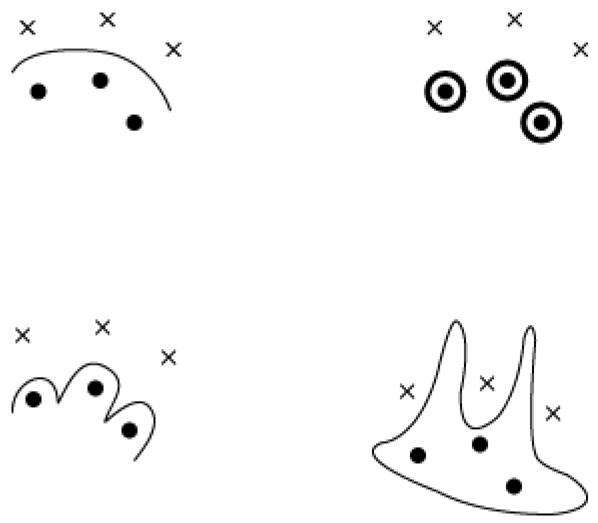

Figure 4.1 Sample decision boundaries separating finite training patterns

In statistical learning theory, it is stated that to find a good decision boundary, regularization [Giroshi F., Jones and M., Poggio T. 1995] is to be applied. In order to produce a unique solution for a learning problem with a set of finite training samples, the dataset need to be constrained through penalizing the complex functions (complex decision boundaries in pattern classification scenario).

It is explained [Bellman R. E., 1961] that the complex decision boundaries often result in a problem called "the curse of dimensionality". Bellman states that "a set of finite training samples imply that any estimate of the unknown classification function is inaccurate (biased). Meaningful estimation is only possible with sufficiently smoothness. For high-dimensional functions, there is an exponential growth in complexity as a result of growth in dimensionality. It is very difficult to collect enough samples to attain the required high density".

Recent studies [Friedman J., 1995] also point out that the basic reason for "curse of dimensionality" is that a function defined in high-dimensional space is likely to be much more complex than a function defined in a lower-dimensional space, and those complications are very hard to discern.

The famous "Occam's razor" principle states the preference of the simple smooth learning models (or decision boundaries in pattern classification scenario) over the complex ones. This is often used as a guideline for us to find the trade-off between the learning model complexity and the learning model accuracy to achieve good generalization performance in a pattern classification system.

In the next sections, we study the pattern space and use "divide and conquer" strategy to divide the pattern space into a few sub-spaces. Instead of using high-dimensional decision boundary to divide the whole input space, less complex decision boundaries are used in the sub-spaces to tackle the classification.

4.2 Pattern Spaces in the S-AdaBoost Algorithm

In order to introduce the S-AdaBoost algorithm, the distribution of the training patterns in the whole input pattern space is outlined. The whole input pattern space $\hat{\mathbf{S}}$ can be denoted by:

$\hat{\mathbf{S}} = \{\mathbf{P} = (\mathbf{X}, \mathbf{Y})\}$

Where,

$\mathbf{X} = \{x_i\}$ denotes the input pattern set;

x_i denotes the number ith input pattern.

$\mathbf{Y} = \{y_i\}$ denotes the classification result set;

y_i denotes the classification result for the number ith input pattern.

$\mathbf{P} = \{p_i = \{(x_i, y_i)\}\}$ denotes the input pattern and the corresponding classification result pair set;

(x_i, y_i) denotes the number ith input pattern and the corresponding classification result pair.

In the S-AdaBoost algorithm, the input training patterns in $\hat{\mathbf{S}}$ can be divided into a few disjoint sub-sets in the corresponding sub-spaces, which fully partition the space $\hat{\mathbf{S}}$ relative to a particular classifier $\mathbf{F(x)}$:

$$\hat{\mathbf{S}} = \hat{\mathbf{S}}_{\mathbf{no}} + \hat{\mathbf{S}}_{\mathbf{sp}} + \hat{\mathbf{S}}_{\mathbf{hd}} + \hat{\mathbf{S}}_{\mathbf{ns}} \qquad (4.2.1)$$

Where,

The "+" means Union;

$\hat{\mathbf{S}}_{\mathbf{no}} = \{\mathbf{P}_{\mathbf{no}}\}$ denotes the sub-space containing the Normal Pattern set (those patterns, which can be easily classified by the classifier $\mathbf{F(x)}$ with a relative simple and smooth decision boundary);

$\mathbf{P}_{\mathbf{no}}$ denotes the Normal Pattern.

$\hat{\mathbf{S}}_{\mathbf{sp}} = \{\mathbf{P}_{\mathbf{sp}}\}$ denotes the sub-space containing the Special Pattern set (those patterns, which can be classified correctly by classifier $\mathbf{F(x)}$ with a not very complex decision boundary);

$\mathbf{P}_{\mathbf{sp}}$ denotes the Special Pattern.

$\hat{\mathbf{S}}_{\mathbf{hd}} = \{\mathbf{P}_{\mathbf{hd}}\}$ denotes the sub-space containing the Hard-To-Classify Pattern set (those patterns, which are hard to be classified by the classifier $\mathbf{F(x)}$ with a relative simple and smooth decision boundary);

$\mathbf{P}_{\mathbf{hd}}$ denotes the Hard-To-Classify Pattern.

$\hat{\mathbf{S}}_{\mathbf{ns}} = \{\mathbf{P}_{\mathbf{ns}}\}$ denotes the sub-space containing the Pattern set with Noise;

$\mathbf{P}_{\mathbf{ns}}$ denotes the Noisy Pattern.

A typical input pattern space is shown in Figure 4.2. If the whole input space $\hat{\mathbf{S}}$ only consists of the normal patterns $\mathbf{P}_{\mathbf{no}}$ from the sub-space $\hat{\mathbf{S}}_{\mathbf{no}}$ (as shown in Figure 4.3), a simple and smooth decision boundary can easily be drawn; a few representative training patterns in the sub-space $\hat{\mathbf{S}}_{\mathbf{no}}$ determine the decision boundary and the margin. If the input space $\hat{\mathbf{S}}$ consists of both the normal patterns $\mathbf{P}_{\mathbf{no}}$ from the sub-space $\hat{\mathbf{S}}_{\mathbf{no}}$ and the special patterns $\mathbf{P}_{\mathbf{sp}}$ from the sub-space $\hat{\mathbf{S}}_{\mathbf{sp}}$ (as shown in Figure 4.4), the decision boundary can still be drawn without losing too much generalization capability, and the margin can also be obtained fairly easily. If the input space $\hat{\mathbf{S}}$ consists of both the normal patterns $\mathbf{P}_{\mathbf{no}}$ from the sub-space $\hat{\mathbf{S}}_{\mathbf{no}}$ and the special patterns $\mathbf{P}_{\mathbf{sp}}$ from the sub-space $\hat{\mathbf{S}}_{\mathbf{sp}}$ as well as the hard-to-classify patterns $\mathbf{P}_{\mathbf{hd}}$ from the sub-space $\hat{\mathbf{S}}_{\mathbf{hd}}$ (as shown in Figure 4.5), the decision boundary can not be drawn without overfitting the training input patterns or sacrificing the generalization capability, much difficulty is needed to draw even a very narrow margin. If the input space $\hat{\mathbf{S}}$ consists of both the normal patterns $\mathbf{P}_{\mathbf{no}}$ from the sub-space $\hat{\mathbf{S}}_{\mathbf{no}}$ and the special patterns $\mathbf{P}_{\mathbf{sp}}$ from the sub-space $\hat{\mathbf{S}}_{\mathbf{sp}}$ as well as the hard-to-classify patterns $\mathbf{P}_{\mathbf{hd}}$ from the sub-space $\hat{\mathbf{S}}_{\mathbf{hd}}$ in a noisy environment with the noisy patterns $\mathbf{P}_{\mathbf{ns}}$ from the sub-space $\hat{\mathbf{S}}_{\mathbf{ns}}$ (as shown in Figure 4.6), the noisy patterns in the sub-space $\hat{\mathbf{S}}_{\mathbf{ns}}$ will affect both the accuracy and the generalization capability of the algorithm, and the decision boundary can hardly be drawn; with overfitting and distortion, the margin is hardly noticeable in this scenario.

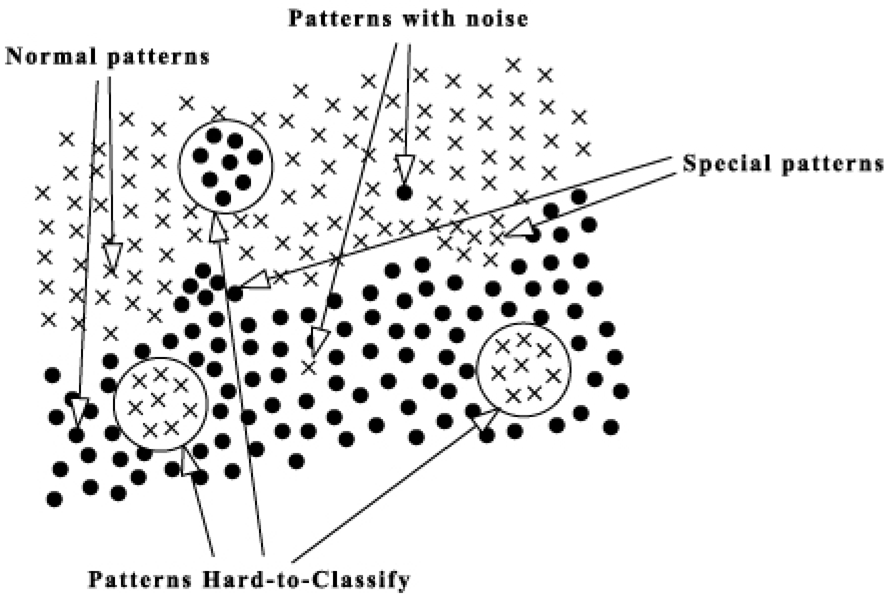

Figure 4.2 Input Pattern Space $\hat{\boldsymbol{S}}$

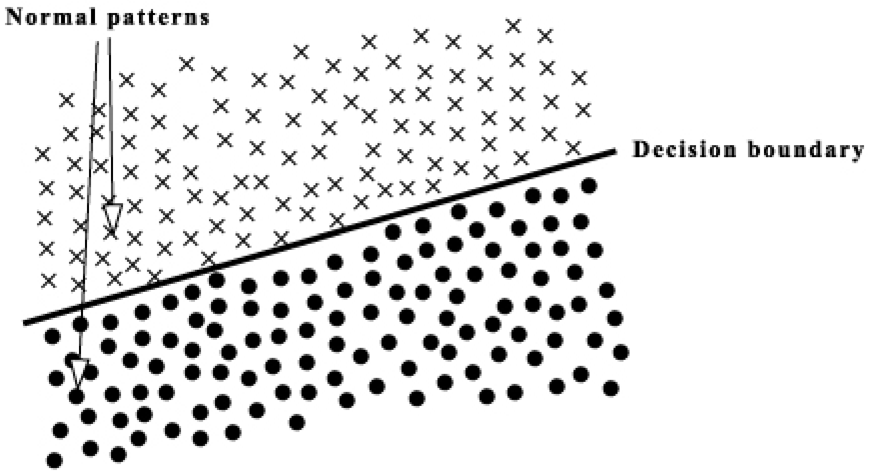

Figure 4.3 Input Pattern Space with normal patterns $\boldsymbol{P}_{no}$

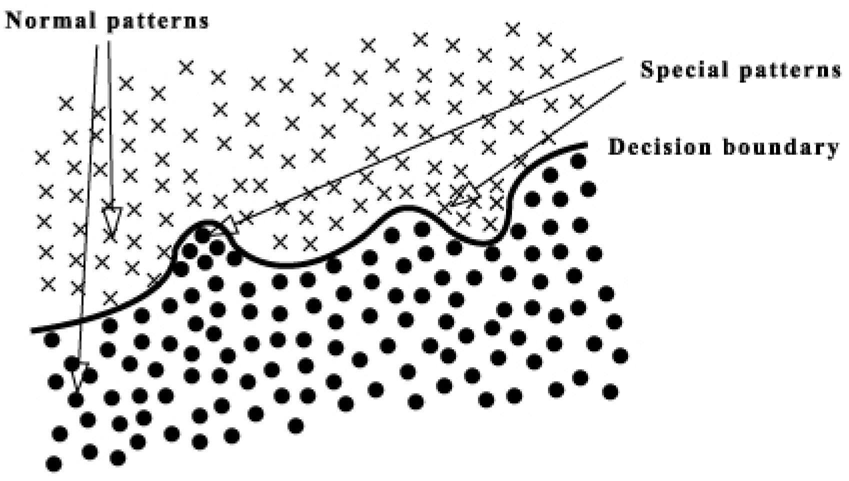

Figure 4.4 Input Pattern Space with normal patterns $\boldsymbol{P_{no}}$ *and special patterns* $\boldsymbol{P_{sp}}$

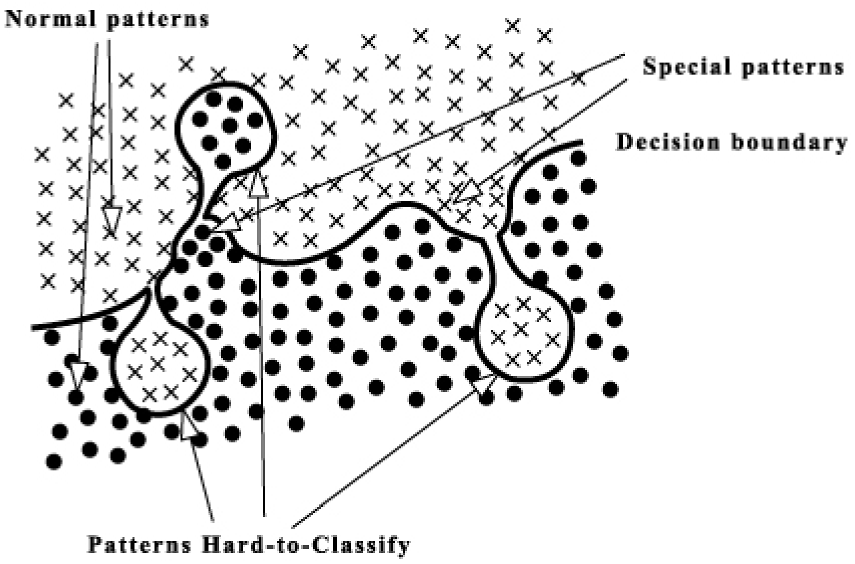

Figure 4.5 Input Pattern Space with normal patterns $\boldsymbol{P_{no}}$*, special patterns* $\boldsymbol{P_{sp}}$ *and hard-to-classify patterns* $\boldsymbol{P_{hd}}$

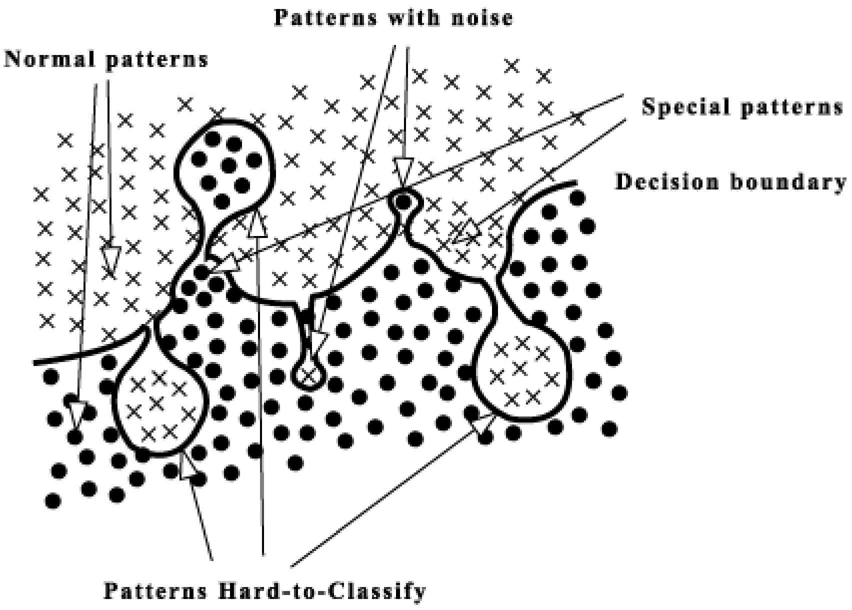

Figure 4.6 Input Pattern Space with normal patterns $\boldsymbol{P_{no}}$, special patterns $\boldsymbol{P_{sp}}$, hard-to-classify patterns $\boldsymbol{P_{hd}}$ and noisy patterns $\boldsymbol{P_{ns}}$

In the S-AdaBoost algorithm, the normal pattern sub-space $\hat{\mathbf{S}}_{\mathbf{no}}$ and the special pattern sub-space $\hat{\mathbf{S}}_{\mathbf{sp}}$ are jointly referred to as the ordinary pattern (or called inlier) sub-space $\hat{\mathbf{S}}_{\mathbf{od}}$, and the hard-to-classify patterns sub-space $\hat{\mathbf{S}}_{\mathbf{hd}}$ and the noisy pattern sub-space are jointly called the outlier sub-space $\hat{\mathbf{S}}_{\mathbf{ol}}$:

$$\hat{S}_{od} = \hat{S}_{no} + \hat{S}_{sp} \tag{4.2.2}$$

$$\hat{\mathbf{S}}_{\mathbf{ol}} = \hat{\mathbf{S}}_{\mathbf{hd}} + \hat{\mathbf{S}}_{\mathbf{ns}} \tag{4.2.3}$$

As shown in Figure 4.6, it is sometimes difficult to classify all the patterns in $\hat{\mathbf{S}}$ well using a single classifier **F(x)** with a simple and smooth decision boundary. Notwithstanding, after dividing the whole input space $\hat{\mathbf{S}}$ into the ordinary pattern sub-space $\hat{\mathbf{S}}_{\mathbf{od}}$ and the outlier sub-space $\hat{\mathbf{S}}_{\mathbf{ol}}$, it is relatively easier for an algorithm like AdaBoost to classify the ordinary pattern sub-space $\hat{\mathbf{S}}_{\mathbf{od}}$ well with a not very complex decision boundary (as shown in Figure 4.4). However, to correctly classify patterns in both the ordinary pattern sub-space $\hat{\mathbf{S}}_{\mathbf{od}}$ and the outlier sub-space $\hat{\mathbf{S}}_{\mathbf{ol}}$ well using only one classifier **F(x)** (as shown in Figure 4.5 and Figure 4.6), the trade-off between the complexity and the generalization capability of the S-AdaBoost algorithm needs to be carefully considered. It is well understood that a more complex $\mathbf{F(x)}$ yields lower training errors yet runs the risk of poor generalization. It has been confirmed by a number of researchers [Dietterich and Kong, 1995; Quinlan, 1996; Jiang, 2001; Dietterich, 2000; Grove and Schuurmans, 1998; Rätsch, 1998] that if a system is to use the AdaBoost algorithm alone to handle classifications in both the ordinary pattern sub-space $\hat{\mathbf{S}}_{\mathbf{od}}$ and the outlier sub-space $\hat{\mathbf{S}}_{\mathbf{ol}}$ well, classifier $\mathbf{F(x)}$ will focus intensively on the noisy patterns $\mathbf{P_{ns}}$ and the hard-to-classify patterns $\mathbf{P_{hd}}$ in the outlier sub-space $\hat{\mathbf{S}}_{\mathbf{ol}}$ and the generalization characteristic of the system will be affected in the complex real world environment.

4.3 The S-AdaBoost Machine

During training, instead of manipulating the whole input space $\hat{\mathbf{S}}$ with the weighted training distribution and using only one model to fit all the training patterns (often contain the outliers) as done in the AdaBoost algorithm, the S-AdaBoost machine uses an AdaBoost $\mathbf{Đ(t)}$ as a divider to divide all the patterns in the original training input space $\hat{\mathbf{S}}$ into two separate sets in two sub-spaces: the ordinary pattern sub-space $\hat{\mathbf{S}}_{\mathbf{od}}$ and the outlier sub-space $\hat{\mathbf{S}}_{\mathbf{ol}}$. The set of patterns in the ordinary pattern sub-space $\hat{\mathbf{S}}_{\mathbf{od}}$ is used to train the next AdaBoost classifier $\mathbf{F_{od}(x)}$, which has a good generalization characteristic; and another set of patterns in the outlier sub-space $\hat{\mathbf{S}}_{\mathbf{ol}}$ is used to train a dedicated outlier classifier $\mathbf{O(x)}$, which has a good localization characteristic. The framework of the S-AdaBoost machine is shown in Figure 4.7. During testing, the divider $\mathbf{Đ(t)}$ is no longer needed, testing patterns are fed directly to the AdaBoost classifiers **Ƒ$_{od}$(x)** and the outlier classifier **O(x)** followed by the combiner **Ĉ** to obtain the classification results.

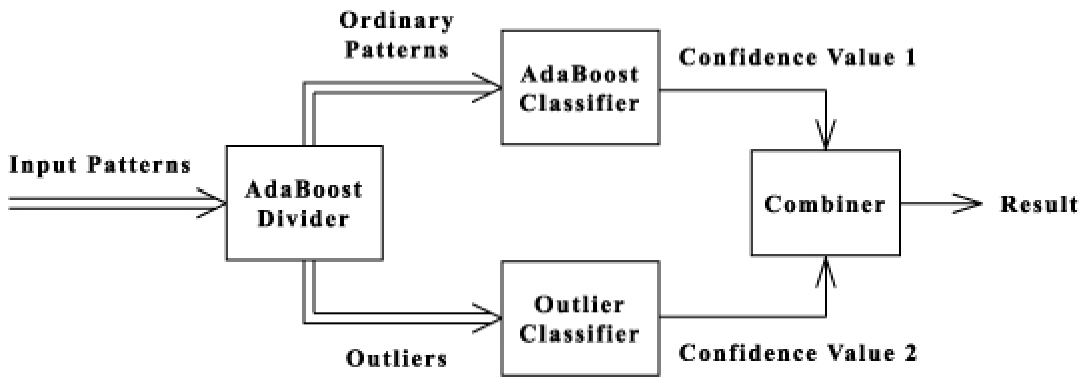

Figure 4.7 The S-AdaBoost Machine in Training

4.4 The Divider of the S-AdaBoost Machine

S-AdaBoost algorithm innovatively uses the AdaBoost's adaptive distributive weight as a dividing tool to divide the input space into sub-spaces. An AdaBoost Đ(ŧ) (as shown in Figure 4.8) is used in the S-AdaBoost machine as a divider to divide the original training set into two separate sets in the ordinary pattern sub-space **Ŝ$_{od}$** and the outlier sub-space **Ŝ$_{ol}$**. The reason for using the AdaBoost algorithm in both the divider **Đ**(ŧ) and the classifier **Ƒ$_{od}$(x)** is to ensure the optimal performance of the classifier **Ƒ$_{od}$(x)**. The pseudo code of the AdaBoost Divider Đ(ŧ) for a two-class classification can be described as follows:

```
Given: Weak learning algorithm Ŵ;

    Training patterns: $\hat{S}$ = P ={$p_i$ = ($x_i$, $y_i$)} for i = 1 to M
        Where M stands for the number of the training patterns;
        $x_i \in X$ stands for the input patterns; and
        $y_i \in Y$ ={-1,1} stands for the targeted output;
    Number of iteration T;
```

```
        The threshold value $\mathbf{t}$.
L0: Initialize the two sub-spaces:
        $\hat{\mathbf{S}}_{\mathbf{od}} = \hat{\mathbf{S}}$;
        $\hat{\mathbf{S}}_{\mathbf{ol}} = \{\}$;
        m=M.
L1: Initialize distribution $D$:
        Set $D_1(i) = \frac{1}{M}$, for all i = 1 to M;
        Set iteration count t = 1;
        Set divide=0;
        Set initial error rate $\in_1 = 0$.
L2: Iterate while $\left(\in_t < 0.5\right)$ and $(t \le T)$
        -  Call $\hat{\mathbf{W}}$ algorithm with distribution $D_i$:
              Obtain the hypothesis $\mathbf{h}_t : X \to Y$
        -  Calculate the weighted error rate:
              $\in_t = \sum_{i:h_t(x_i) \neq y_i} D_t(i)$
        -  Set $\beta_t = \frac{\in_t}{\left(1 - \in_t\right)}$
        -  Update the new distribution D for i= 1 to M:
              $D_{t+1}(i) = \frac{D_t(i)\beta_t^{\mathrm{Sign}(h_t(x_i) == y_i)}}{z_t}$;
        -  Where $zt$ is a normalization factor chosen such that the new distribution
           $D_{t+1}$ is a normalized distribution, where $\mathrm{Sign}(x) = \begin{cases} 1 & \text{if } X > 0 \\ 0 & \text{if } X \le 0 \end{cases}$
        -  t = t +1;
        -  For $i$=1 to m,
              BEGIN
                    If ($D_t(i)$ > the threshold value $\mathbf{t}$)
                    BEGIN
```

```
                m =m-1;
                $\hat{\mathbf{S}}_{\mathbf{od}} = \hat{\mathbf{S}}_{\mathbf{od}} - \{P_i\}$; (Set difference)
                $\hat{\mathbf{S}}_{\mathbf{ol}} = \hat{\mathbf{S}}_{\mathbf{ol}} + \{P_i\}$; (Set Union)
                divide =1
            END
            If (divide=1)
            Go to L1.
        END
L3:  Export the ordinary pattern sub-space $\hat{\mathbf{S}}_{\mathbf{od}}$ and the outlier sub-space $\hat{\mathbf{S}}_{\mathbf{ol}}$
```

From the above algorithm description, it is noticed that:

$$\hat{S}_{od} \cup \hat{S}_{ol} = \hat{S} \tag{4.5.1}$$

$$\hat{\mathbf{S}}_{od} \cap \hat{\mathbf{S}}_{ol} = \Phi \tag{4.5.2}$$

As the inlier and outlier sub-spaces defined above are relative to a particular classifier, they are not absolute concepts and they change with the selection of different threshold values. According to the above S-AdaBoost algorithm, the initial value of the Inlier sub-space is defined as the Input space, and the initial value of the Outlier sub-space is defined as empty. Only after an area is classified as the Outlier sub-space, the area is removed from the Inlier sub-space, this ensures that the two sub-spaces cover the whole Input space and the two sub-spaces do not overlap with each other. It is noticed that initially all patterns belongs to $\hat{\mathbf{S}}_{od}$, if in an iteration, the distribution weight of a particular pattern $\mathbf{p}_i$ exceeds the threshold, the pattern $\mathbf{p}_i$ is removed from the $\hat{\mathbf{S}}_{od}$ and placed into $\hat{\mathbf{S}}_{ol}$, this practice ensures that the two sub-spaces are mutually exclusive.

After running the above codes, the patterns in the whole input space $\hat{\mathbf{S}}$ will be divided into two sets: the ordinary pattern sub-space $\hat{\mathbf{S}}_{\mathbf{od}}$ and the outlier sub-space $\hat{\mathbf{S}}_{\mathbf{ol}}$.

Choosing the optimal threshold value **t** is task specific. It will be discussed in the following sections.

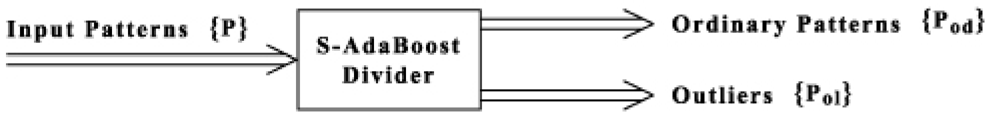

Figure 4.8 The Divider of the S-AdaBoost Machine

4.5 The Classifiers in the S-AdaBoost Machine

After dividing the training patterns in the whole input space $\hat{\mathbf{S}}$ into the ordinary pattern sub-space $\hat{\mathbf{S}}_{od}$ and the outlier sub-space $\hat{\mathbf{S}}_{ol}$, the normal patterns $\mathbf{P}_{no}$ and the special patterns $\mathbf{P}_{sp}$ in the ordinary pattern sub-space $\hat{\mathbf{S}}_{od}$ are used to train the $\mathbf{F}_{od}(\mathbf{x})$ classifier, whereas the noisy patterns $\mathbf{P}_{ns}$ and the hard-to-classify patterns $\mathbf{P}_{hd}$ in the outlier sub-space $\hat{\mathbf{S}}_{ol}$ are used to train the outlier classifier $\mathbf{O}(\mathbf{x})$ in the S-AdaBoost Machine.

After certain rounds of iteration, the classifier $\mathbf{F}_{od}(\mathbf{x})$ focuses more on the *difficult* special patterns $\mathbf{P}_{sp}$ and less on the *easy* normal patterns $\mathbf{P}_{no}$ in forming the decision boundary. As the special patterns $\mathbf{P}_{sp}$ are not outliers, the accuracy and the generalization properties of the classifier $\mathbf{F}_{od}(\mathbf{x})$ are not affected. Making use of the randomness nature of the noisy patterns $\mathbf{P}_{ns}$, $\mathbf{O}(\mathbf{x})$, a classifier preferably with the good localization characteristics, can identify the local clustering of the hard-to-classify patterns $\mathbf{P}_{hd}$ and at the same time isolate the noisy patterns $\mathbf{P}_{ns}$ from the hard-to-classify patterns $\mathbf{P}_{hd}$. The pseudo code of the classifier $\mathbf{F}_{od}(\mathbf{x})$ can be described as follows:

```
Given: Weak learning algorithm $\hat{\mathbf{W}}$;
    Training patterns: $\hat{\mathbf{S}}_{od} = \{\mathbf{P}\} = \{\mathbf{p}_i = (\mathbf{x}_i, \mathbf{y}_i)\}$ for i = 1 to M
```

```
        Where M stands for the number of the training patterns in $\hat{\mathbf{S}}_{\mathbf{od}}$;
        $\mathbf{x}_i \in \mathbf{X}$ stands for the input patterns;
        and $\mathbf{y_i} \in \mathbf{Y} = \{-1,1\}$ stands for the targeted output;
    Number of iteration T;
L0: Initialize the parameter:
    m=M.
L1: Initialize distribution $D$:
    Set $D_1(i) = \frac{1}{M}$, for all i = 1 to M;
    Set iteration count t = 1;
    Set initial error rate $\in_1 = 0$.
L2: Iterate while $(\in_t < 0.5)$ and $(t \le T)$
    -   Call $\hat{\mathbf{W}}$ algorithm with distribution $D_i$:
            Obtain the hypothesis $\mathbf{h}_t : X \to Y$
    -   Calculate the weighted error rate:
            $\in_t = \sum_{i:h_t(x_i) \neq y_i} D_t(i)$
    -   Set $\beta_t = \frac{\in_t}{(1 - \in_t)}$
    -   Update the new distribution D for i= 1 to M:
            $D_{t+1}(i) = \frac{D_t(i)\beta_t^{Sign(h_t(x_i) == y_i)}}{z_t}$;
        Where zt is a normalization factor chosen such that the new distribution
        Dt+1 is a normalized distribution, where $Sign(x) = \begin{cases} 1 & \text{if } X > 0 \\ 0 & \text{if } X \le 0 \end{cases}$
    -   t = t +1;
L3:     Calculate the classification confidence value
```

$$\underset{y \in Y}{\text{Max}} \sum_{t=1,\, h_t(x)=y}^{T} \text{Log}\left(\frac{1}{\beta_t}\right)$$

The base classifier of the AdaBoost Divider and Inlier Classifier can be implemented using the neural network algorithm, C4.5 tree algorithm or some linear classifier. The right choice is depending on the classification task, the complexity of the input patterns and other factors. In the thesis, some common classifiers are implemented to evaluate the influence of the base classifier to the overall performance of the system.

The requirement to the outlier classifier **O(x)** is that it must be able to localize the influence of the patterns in the outlier sub-space $\hat{\mathbf{S}}_{\mathbf{ol}}$ (as shown in Figure 4.9). RBF Neural Network [Chen and Liu, 1992] is used to implement the outlier classifier in the thesis.

It is understood that AdaBoost is distance based classifier maximizing the margin. Intuitively, when outliers are handled by a separate outlier classifier **O(x)**, the margin of the system is preserved or increased.

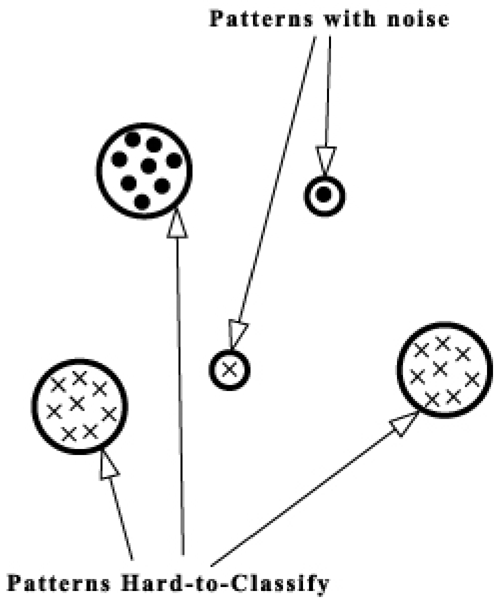

Figure 4.9 Localization of the Outlier Classifier ***O(x)*** *in the S-AdaBoost machine*

4.6 The Combiner and the complexity of the S-AdaBoost Machine

Many combination methods [Tresp and Maniguchi, 1995; Jacobs, Jordan, Nowlan and Hinton, 1991; Jordan and Jacobs, 1994] are available for selection. Noticing that the classifiers $\mathbf{F}_{\mathbf{od}}\mathbf{(x)}$ and $\mathbf{O(x)}$ are of different structure and nature, a non-linear combiner $\hat{\mathbf{C}}$ instead of linear ones like averaging, voting and Borda count [Ho, Hull and Srihari, 1994] is used to combine the classification results of the $\mathbf{F}_{\mathbf{od}}\mathbf{(x)}$ and $\mathbf{O(x)}$. Nevertheless, in the following sections, different combiners are implemented to support the intuitive selection. The confidence values generated by the S-AdaBoost classifiers act as the input and the final classification result are the output of the combiner $\hat{\mathbf{C}}$.

The system complexity of S-AdaBoost in training can be denoted as the sum of the complexity of AdaBoost Divider Đ(ŧ), the maximum of the complexity of the Inlier Classifier $\mathbf{F}_{\mathbf{od}}\mathbf{(x)}$ and the Outlier Classifier $\mathbf{O(x)}$, and the complexity of the Combiner $\hat{\mathbf{C}}$:

$$O_{train}(\text{S-AdaBoost}) = O(\text{Đ}(\text{ŧ})) + \text{Max}\,[O(\mathbf{F}_{\mathbf{od}}\mathbf{(x)}), O(\mathbf{O(x)})] + O(\hat{\mathbf{C}}) \quad (4.6.1)$$

Since the Outlier Classifier **O(x)** is implemented by a simple RBF Neural Network, whereas the $\mathbf{F_{od}(x)}$ is a mixture of the classifier, the complexity of the $\mathbf{F_{od}(x)}$ is generally much higher than that of **O**(x), we have:

$$\text{Max}\ [O(\mathbf{F_{od}(x)}), O(\mathbf{O(x)})] = O(\mathbf{F_{od}(x)}) \qquad (4.6.2)$$

The Combiner $\hat{\mathbf{C}}$ is normally implemented by a simple Multilayer Perceptron or some simpler mechanism, we have:

$$O(\hat{\mathbf{C}}) =< O\ (Đ(\mathbf{t})) \qquad (4.63)$$

As mentioned before, the AdaBoost Divider Đ(**t**) and the Inlier Classifier $\mathbf{F_{od}(x)}$ are of the same structure, we have:

$$O\ (Đ(\mathbf{t})) = O(\mathbf{F_{od}(x)}) \qquad (4.6.4)$$

Combining (4.6.1), (4.6.2), (4.6.3), and (4.6.4), we conclude:

$$\begin{aligned} O_{train}\ (\text{S-AdaBoost}) &= O\ (Đ(\mathbf{t})) + \text{Max}\ [O(\mathbf{F_{od}(x)}), O(\mathbf{O(x)})] + O(\hat{\mathbf{C}}) \\ &= O\ (Đ(\mathbf{t})) + O(\mathbf{F_{od}(x)}) + O(\hat{\mathbf{C}}) \\ &= 2O\ (Đ(\mathbf{t})) + O(\hat{\mathbf{C}}) \\ &<= 3\ O\ (Đ(\mathbf{t})) \qquad (4.6.5) \end{aligned}$$

The above simple analysis shows that the complexity of S-AdaBoost in training is still not very significant complex comparing with that of an AdaBoost.

Similarly, the system complexity of S-AdaBoost in testing can be denoted as the sum of the maximum of the complexity of the Inlier Classifier $\mathbf{F_{od}(x)}$ and the Outlier Classifier **O(x),** and the complexity of the Combiner $\hat{\mathbf{C}}$:

$$O_{testing}(\text{S--AdaBoost}) = \text{Max}\ [O(\mathbf{F_{od}(x)}), O(\mathbf{O(x)})] + O(\hat{\mathbf{C}}) \qquad (4.6.6)$$

Combining (4.6.2), (4.6.3), (4.6.6), we have:

$$\begin{aligned} O_{testing}\ (\text{S-AdaBoost}) &= \text{Max}\ [O(\mathbf{F_{od}(x)}), O(\mathbf{O(x)})] + O(\hat{\mathbf{C}}) \\ &= O(\mathbf{F_{od}(x)}) + O(\hat{\mathbf{C}}) \\ &= O\ (Đ(\mathbf{t})) + O(\hat{\mathbf{C}}) \\ &<= 2\ O\ (Đ(\mathbf{t})) \qquad (4.6.7) \end{aligned}$$

The above simple analysis also shows that the complexity of S-AdaBoost in testing is still not very significant complex comparing with that of an AdaBoost.

4.7 Statistical analysis of the S-AdaBoost learning

S-AdaBoost's effectiveness can be analyzed through bias/variance analysis. It is proven (see appendix for detail) that by dividing a problem into sub-problems in whole or sub-spaces, and using ensemble boosting approach, the overall performance can be boosted.

According to the statistical learning theory (Vapnik, 1995), in the neural network environment, the learning is a process focusing on reducing the deviation between a targeted function and the real function.

In statistical learning theory, the mean squared error is equal to the sum the variance and square of the bias. Bias denotes the inability of the individual component of the ensemble to accurately measure the decision boundary; Variance denotes the inadequacy and the limitation of the training pattern set. Differentiating these two kinds of errors will help us to analyze the effectiveness of the ensemble machines. It is noticed that in a single neural network system with a fixed size of training set, if we want to reduce bias, the variant often goes up; if we want to reduce variant, we need to pay the price of bias going up. It is only possible to reduce both bias and variance when we have infinite number of good training patterns (Geman, Bienenstock and Doursat, 1992), this phenomenon is called the bias/variance dilemma. To tackle this dilemma, at least two approaches have been proposed. One way to circumvent this is to introduce "harmless" bias to maintain bias and reduce variance [LeCun 1990], another way is to maintain bias and reduce variance using ensemble approaches

[Naftaly, Intrator and Horn,1997]. The S-AdaBoost ensemble method falls into the second category.

In the S-AdaBoost algorithm based classifiers, differently trained classifiers (we use the neural network classifiers in this section for discussion) share different distributions of the training input patterns; the classification results (confidence values) of these classifiers are combined to produce the final classification output. It is noticed that if all those individual classifiers were combined to form one big classifying neural network, the number of free parameters to be decided would be numerous, which could easily lead to the overfitting of the big network. In the ensemble approach, each component classifier is trained separately; the chance of overfitting is thus reduced.

In the ensemble approach, it is expected that each hypothesis classifier focuses on its own domain and converges to its own local minimum of the cost function. It is hoped that in this way the combiner can boost the combined performance of the ensemble.

As proven in Appendix A, for a simple ensemble machine with averaging combination, we conclude that the ensemble can help to reduce the overall error rate. In the S-AdaBoost machine, the classifiers are extended to different types and the combination method is expanded to be non-linear, which can further regulate the bias/variance trade-off and error rate.

4.8 Choosing the Threshold Value $\mathbf{t}$ in the S-AdaBoost Machine

The threshold value $\mathbf{t}$ is an important non-adaptive parameter in the S-AdaBoost algorithm that needs to be decided to make the algorithm adaptive. The optimal value of the threshold value $\mathbf{t}$ in the Divider Đ($\mathbf{t}$) dividing network is associated with the classification task itself and the nature of the input pattern set**.** According to statistical learning theory, many constrains exist in a classifier. Two of the constrains are: the accuracy of the training input patterns and the limitation of the number of training patterns. As the outliers in the training pattern set may not be able to be inferred easily, some extra argument must be used to describe the classification process, which means:

$$\mathbf{y} = f(\mathbf{x}) + \boldsymbol{\varepsilon} \qquad (4.8.1)$$

Where both **y** and **x** are variables; and the variable **y** statistically depends on the variable **x**. The dependency between the variable **x** and the variable **y** can be learned, but the causality between the variable **x** and the variable **y** can not be inferred from the data alone. Priori empirical knowledge can sometimes help to find the causality between the variable **x** and the variable **y** together with the limited non-perfect training data. One of the well-known principles is the "Occam's razor" principle, which states the preference of the simple learning models over the complex ones; this is often used to guide us to find the trade-off between the learning model complexity and the learning model accuracy to achieve good generalization performance of the learning system. If some outliers, which include the noisy patterns and the hard to be classified patterns, are encountered; some dividing mechanisms are explored in the thesis instead of using a complex learning model to describe the unknown causality in the learning model.

The inductive principles are used to optimize the above trade-off, the threshold value $\mathbf{t}$ can be understood as one of the regularization parameters used in the S-AdaBoost based systems for good generalization performance.

According to the Penalization (Regularization) Inductive principle, the classification problem can be understood as a class of approximating functions $\mathbf{f}_i$s:

$\mathbf{y}_i = \mathbf{f_i}\ (\mathbf{x}, \boldsymbol{\varphi})$
Where $\mathbf{x} \in \mathbf{X}$; and
$\boldsymbol{\varphi} \in \boldsymbol{\Psi}$, which denotes the set of free parameters.

To find the optimal function among the function class, a penalization (regularization) term is added to the following empirical risk item to be minimized:

$$\acute{\mathbf{R}}_{\text{reg},\mathbf{i}}\ (\boldsymbol{\varphi}) = \acute{\mathbf{R}}_{\text{emp},\mathbf{i}}\ (\boldsymbol{\varphi}) + \boldsymbol{\lambda}\ \acute{\boldsymbol{\emptyset}}\ [\mathbf{f_i}(\mathbf{x}, \boldsymbol{\varphi})] \qquad (4.8.2)$$

Where, $\acute{\mathbf{R}}_{\text{reg},}\mathbf{i}\ (\boldsymbol{\varphi})$ denotes the regularization risk; and
$\acute{\mathbf{R}}_{\text{emp},}\mathbf{i}\ (\boldsymbol{\varphi})$ denotes the empirical risk;

$\acute{\boldsymbol{\emptyset}}\ [\mathbf{f_i}(\mathbf{x}, \boldsymbol{\varphi})]$ denotes the regularization term, which is a non-negative function associated with each approximating function $\mathbf{f_i}(\mathbf{x}, \boldsymbol{\varphi})$; it is often assigned to be small values for the smooth functions and large values for the non-smooth functions;

Parameter $\boldsymbol{\lambda}$ is called the regularization parameter, which is a positive number controlling the strength of the regularization.

The priori empirical knowledge is embedded in the regularization term and the strength of the influence of the regularization is determined by the parameter $\boldsymbol{\lambda}$. In the S-AdaBoost algorithm, the regularization is implemented by the Divider ($\boldsymbol{\mathfrak{t}}$), the scale of the regularization is implemented by the parameter $\boldsymbol{\mathfrak{t}}$. If $\boldsymbol{\lambda}$ is chosen to be a large number, $\acute{\mathbf{R}}_{\text{reg},}\mathbf{i}\ (\boldsymbol{\varphi})$ depends largely on the regularization term $\boldsymbol{\lambda}\ \acute{\boldsymbol{\emptyset}}\ [\mathbf{f_i}(\mathbf{x}, \boldsymbol{\varphi})]$; if $\boldsymbol{\lambda}$ is chosen to be a small number, $\acute{\mathbf{R}}_{\text{reg},}\mathbf{i}\ (\boldsymbol{\varphi})$ depends largely on the regularization term $\boldsymbol{\lambda}\ \acute{\boldsymbol{\emptyset}}$ $[\mathbf{f_i}(\mathbf{x}, \boldsymbol{\varphi})]$. For a given regularization term $\acute{\boldsymbol{\emptyset}}\ [\mathbf{f_i}(\mathbf{x}, \boldsymbol{\varphi})]$, the complexity of the learning model is controlled by the regularization parameter $\boldsymbol{\lambda}$. The optimal value of $\boldsymbol{\lambda}$ (which can help to generate the minimum regularization risk) is usually determined empirically (as the case of parameter of $\mathbf{t}$ in the Divider ($\mathbf{t}$)). Experiments are conducted to find a formula to decide the optimal value of $\mathbf{t}$ in the S-AdaBoost based face detection FDAO system and face identification FISA system as well as the GMD benchmark databases.

The threshold value $\mathbf{t}$ plays a very important role in the S-AdaBoost machine. It is noticed that: If the threshold value $\mathbf{t}$ is less than or equal to value 0, $\hat{\mathbf{S}}_{\mathbf{od}}$ is empty; all the patterns in $\hat{S}$ are treated as outliers; the S-AdaBoost machine becomes a large memory network; and the Outlier Classifier $\mathbf{O}(\mathbf{x})$ determines the performance of the S-AdaBoost machine. If the threshold value $\mathbf{t}$ is greater than or equal to value 1, $\hat{\mathbf{S}}_{\mathbf{ol}}$ is empty; no patterns in the Input Space $\hat{\mathbf{S}}$ are treated as outliers; the S-AdaBoost machine becomes an AdaBoost machine; and the performance of the S-AdaBoost machine is determined by the AdaBoost Classifier $\mathbf{F}_{\mathbf{od}}(\mathbf{x})$ itself.

The AdaBoost machine can be considered as a special implementation of the S-AdaBoost machine when the threshold value $\mathbf{t}$ is greater than or equal to 1. In the coming discussion, the parameter δ is defined as the S-AdaBoost machine's Error Rate when the threshold value $\mathbf{t}$ is equal to value 1.

In practical systems handling real world scenario, the special patterns sometimes can be misclassified as outliers when the threshold value $\mathbf{t}$ increases, this will not affect the classification result very much in S-AdaBoost, these patterns are just not participating in the forming of the decision boundary in the Ordinary Space, they can still contribute to the final classification result from the Outlier Space, the Combiner can help to balance the confidence values from the two sub-classifiers.

The optimal value of the threshold value $\mathbf{t}$ is associated with the classification task itself and the nature of the input patterns in the whole input space $\hat{\mathbf{S}}$. Experiments are conducted to determine the optimal value chosen for the threshold value $\mathbf{t}$. From the experiments conducted, as a guideline, the S-AdaBoost machine performs reasonably well when the value of the threshold $\mathbf{t}$ is around $1/(M \text{ X } \delta^2)$, where M is the number of the training patterns and the parameter δ is the error rate (the detection error rate for the face detection and the face identification applications) of the AdaBoost machine when the threshold value $\mathbf{t} = 1$.

Hindsight analysis shows that:

$$1/(M \text{ X } \delta^2) = (1/M \text{ X } \delta)\ (1/\delta). \qquad (4.8.3)$$

As M X δ denotes the number of wrong classifications of the classifier; the threshold is disproportionate to the number of error patterns and the error rate of the classifier. The noisier the pattern space, the smaller the threshold, which means more patterns are classified as outliers.

4.9 Experimental Results on the Benchmark Databases

As pointed out [Dietterich, 1997] that "fundamental research in Machine Learning is inherently empirical", extensive experiments are conducted to test the effectiveness of the S-AdaBoost algorithm-based system on the GMD benchmark databases [GMD] compared with the experimental results from the systems based on the AdaBoost algorithm, the Support Vector Machine algorithm and the Regularized AdaBoost algorithm. Back-propagation neural network is used as the base classifier to implement the AdaBoost algorithms for both the Divider $Đ(\mathbf{t})$ and the Classifier $\mathbf{F_{od}(x)}$

of the S-AdaBoost Machine. The RBF neural network is used to implement the Outlier Classifier **O(x)** of the S-AdaBoost Machine and a three layer back-propagation neural network with 2 input nodes and 2 hidden nodes is used to implement the Combiner $\hat{\mathbf{C}}$ of the S-AdaBoost Machine. The threshold value ŧ in the Divider Đ(ŧ) of the S-AdaBoost Machine is set to be equal to 1/(M X δ^2), where M is the number of all the training patterns and the parameter δ is the error rate of the AdaBoost machine. More implementations of the S-AdaBoost algorithm in the complex airport environment will be discussed in more detail in the following chapters.

The benchmark GMD database include the datasets collected from the UCI [UCI], the DELVE [DELVE] and the STATLOG [STATLOG] benchmark repositories. The original version of this GMD database has been used by Rätsch and other researchers [Weston, 1999; Herbrich and Weston, 1999; Chapelle, Vapnik, and Weston, 2000; P´erez-Cruz, Alarc´on-Diana, Navia-V´azquez and Art´es-Rodr´iguez., 2001]. This collection is a well-balanced mixture of different learning tasks and a mixture of different levels of noise of the data. The numbers of patterns available are also varied among the datasets. As not all problems are binary, classification among sub-classes is performed first to change them into binary problems. In order to conduct some statistical analysis and make full use of the training data, 10-fold cross validation method is used in the experiments. Back-propagation multi-layer neural network is used in the experiments as the base classifier for the AdaBoost Machine, the Regularized AdaBoost Machine. The RBF Kernel is used to implement the SVM Machine. The maximum number of classifiers in the ensemble is set to 40 due to Back-Propagation neural network's relative strong classification performance. For easy comparison, the principles used by Rätsch in his experiments [Rätsch, Onoda and

Müller, 2001] are used to set other non-adaptive parameters in all the machines other than the S-AdaBoost Machine.

In the experiment, the datasets' sample sizes and number of input and output features are listed in Table 4.1. All the training data in all datasets are normalized with a mean of 0 and a standard deviation of 1.

	No. of training samples	No. of testing samples	No. of input features	No. of output features
Ringnorm	4000	3400	20	1
Twonorm	4000	3400	20	1
Image	1200	1100	18	1
Thyroid	100	115	5	1
Splice	1500	1675	60	1
Waveform	2000	2600	21	1
Banana	100	177	9	1
Heart	100	170	13	1
Titanic	1000	1051	3	1
Diabetes	400	368	8	1
German	450	550	20	1
B. Cancer	2000	3300	2	1
F. Solar	500	566	9	1

Table 4.1: Datasets used in the experiment

The test results from the implementations of the leading algorithms (the AdaBoost algorithm, the Support Vector Machine algorithm (RBF Kernel), the Regularized AdaBoost algorithm [Rätsch, Onoda and Müller, 2001]) and the S-

AdaBoost algorithm are listed in Table 4.2. The mean and variance values for the error rates are listed in Table 4.2 based on the 10-fold cross validation calculation.

	AdaBoost	SVM	Reg. AdaBoost	S-AdaBoost
Ringnorm	1.9 ± 0.4	1.6 ± 0.2	1.6 ± 0.1	1.7 ± 0.2
Twonorm	3.0 ± 0.2	2.7 ± 0.2	2.7 ± 0.3	**2.7** ± 0.2
Image	2.9 ± 0.9	2.8 ± 0.5	**2.7** ± 0.4	2.7 ± 0.5
Thyroid	4.5 ± 2.1	4.9 ± 1.8	4.6 ± 2.0	**4.3** ± 2.0
Splice	10.4 ± 1.1	10.6 ± 0.7	9.5 ± 1.0	**9.3** ± 0.8
Waveform	10.6 ± 1.3	9.8 ± 1.3	9.8 ± 1.1	**9.6** ± 1.0
Banana	10.8 ± 0.8	11.0 ± 0.7	10.9 ± 0.7	**10.6** ± 0.5
Heart	20.8 ± 3.2	16.4 ± 3.2	16.5 ± 3.3	**15.9** ± 3.1
Titanic	23.1 ± 1.4	22.2 ± 1.2	22.6 ± 1.2	**22.2** ± 1.1
Diabetes	26.8 ± 2.0	23.7 ± 2.0	23.8 ± 2.3	**23.5** ± 1.6
German	27.5 ± 2.4	22.8 ± 2.0	24.3 ± 2.3	23.8 ± 2.4
B. Cancer	30.8 ± 4.0	26.3 ± 4.5	26.5 ± 4.3	**26.1** ± 4.3
F. Solar	35.7 ± 1.6	32.0 ± 1.6	34.2 ± 1.8	**31.6** ± 1.8
Average Mean	16.1	14.5	14.6	14.1

Table 4.2: Comparison of the error rates among various methods on the benchmark databases.

The table shows that the S-AdaBoost machine performs the best in terms of general performance and achieves the best results in 10 out of 13 tests. Further analysis shows that the S-AdaBoost machine out-performs the AdaBoost machine in all tests, which clearly demonstrate S-AdaBoost's superior performance than AdaBoost. The S-AdaBoost machine also shows its effectiveness in this round of tests by out-performing the SVM machine and the Regularized AdaBoost machine, which are the leading approaches in handling a complex environment. It is also shown that the AdaBoost machine performs reasonably well in low noise applications like the Twonorm, the Thyroid and the Image applications compared with the results of the other machines; at the same time, the performance of S-AdaBoost is shown comparing with results of the other machines in high noise applications like the Cancer, Diabetes, German and F. Solar applications.

Further tests are conducted to test the effectiveness of the S-AdaBoost algorithm using different base classifiers in the AdaBoost Divider Đ(ŧ) and Classifier. AdaBoost $\mathbf{F_{od}(x)}$ machines based on the tree structure based algorithm C4.5 and Linear single layer Perceptron algorithm are also implemented. The detail experimental results are shown in Table 4.3.

	AdaBoost (BP)	S-AdaBoost (C4.5)	S-AdaBoost (Single layer perceptron)	S-AdaBoost (BP)
Ringnorm	1.9 ± 0.4	1.8 ± 0.2	**1.7** ± 0.1	1.7 ± 0.2
Twonorm	3.0 ± 0.2	**2.7** ± 0.2	**2.7** ± 0.2	**2.7** ± 0.2
Image	2.9 ± 0.9	**2.5** ± 0.7	2.8 ± 0.3	2.7 ± 0.5
Thyroid	4.5 ± 2.1	**4.3** ± 1.8	4.6 ± 2.4	4.3 ± 2.0
Splice	10.4 ± 1.1	9.3± 1.1	9.5 ± 1.6	**9.3** ± 0.8
Waveform	10.6 ± 1.3	**9.4**± 1.3	9.8 ± 1.4	9.6 ± 1.0
Banana	10.8 ± 0.8	10.9± 0.7	**10.5**± 0.8	10.6 ± 0.5
Heart	20.8 ± 3.2	16.3 ± 3.3	17.3 ± 3.6	**15.9** ± 3.1
Titanic	23.1 ± 1.4	**22.2** ± 1.0	23.0 ± 1.6	22.2 ± 1.1
Diabetes	26.8 ± 2.0	23.7 ± 2.0	24.8 ± 2.3	**23.5** ± 1.6
German	27.5 ± 2.4	24.5 ± 2.3	25.6 ± 2.1	**23.8** ± 2.4

B. Cancer	30.8 ± 4.0	27.9 ± 4.0	27.0 ± 4.7	**26.1** ± 4.3
F. Solar	35.7 ± 1.6	**31.3** ± 1.8	32.1 ± 1.7	31.6 ± 1.8
Average Mean	16.1	14.4	14.7	14.1

Table 4.3: Comparison of the error rates among different base classifier based S-AdaBoost classifiers on the benchmark databases.

From Table 4.3, it is shown that generally, the performance of S-AdaBoost machines based on C4.5, linear Single Layer Perceptron, BP Neural Network achieve better results than AdaBoost machine. The S-AdaBoost machines based on C4.5 and BP Neural Network achieve slight better results than S-AdaBoost machine based on linear Single Layer Perceptron. It is also observed that the three S-AdaBoost machines achieve similar results in low noise applications, but Linear Single layer Perceptron based S-AdaBoost machine performs not as good as the other two in high noise applications. This might explains why C4.5 and BP are the two most popular base classifiers for AdaBoost based systems. As the three S-AdaBoost machines universally perform better than the original AdaBoost machine, the experiment shows that S-AdaBoost is superior in performance than the AdaBoost algorithm irregardless of the base classifier used in our experiment; the choice of base classifier generally does not affect the system performance very much.

Table 4.4 compares the mean errors of S-AdaBoost based on Single Layer Perceptron (Error 2) with the AdaBoost (Error 1) and S-AdaBoost based on BP Neural Network (Error 3). Volume 5 shows how much better the S-AdaBoost based on single layer Perceptron comparing with the AdaBoost Machine, volume 6 demonstrates how much better the S-AdaBoost based on BP neural Network comparing with the S-AdaBoost based on single layer Perceptron.

	Error 1: AdaBoost (BP)	Error 2: S-AdaBoost (Single layer perceptron)	Error 3: S-AdaBoost (BP)	Comparison between Error 2 and Error 1 (betterment %)	Comparison between Error 3 and Error 2 (betterment %)
Ringnorm	1.9	**1.7**	**1.7**	*10.5*	*0*
Twonorm	3.0	**2.7**	**2.7**	*10*	*0*
Image	2.9	2.8	**2.7**	*3.4*	*3.6*
Thyroid	4.5	4.6	**4.3**	*-2.2*	*6.5*
Splice	10.4	9.5	**9.3**	*8.7*	*2.1*
Waveform	10.6	9.8	**9.6**	*7.5*	*2.0*
Banana	10.8	**10.5**	10.6	*2.8*	*-0.9*
Heart	20.8	17.3	**15.9**	*16.9*	*8.1*
Titanic	23.1	23.0	**22.2**	*0.4*	*3.5*
Diabetes	26.8	24.8	**23.5**	*7.5*	*5.2*
German	27.5	25.6	**23.8**	*6.9*	*7*
B. Cancer	30.8	27.0	**26.1**	*12.3*	*3.3*
F. Solar	35.7	32.1	**31.6**	*10.1*	*1.6*
Average Mean	16.1	14.7	**14.1**	*8.7*	*4.1*

Table 44: Comparison of the mean error rates between the linear Single Layer Perceptron and the non-linear BP based S-AdaBoost classifiers on the benchmark databases.

It is observed from Table 4.4 that the linear Single Layer Perceptron based S-AdaBoost machine achieves better results comparing with the AdaBoost machine (in 12 of the 13 cases); the average betterment is 8.7%. The non-linear BP based S-AdaBoost machine achieves better results comparing with the linear Single Layer Perceptron based S-AdaBoost machine (in 10 of the 13 cases); the average betterment is 4.1%. Generally there is no clear indications showing in which "noise level", the non-linear BP based S-AdaBoost machine has a overwhelming advantage over the linear Single Layer Perceptron based S-AdaBoost machine, except showing statistically that the non-linear one is better in overall results.

Different Combiners are implemented to understand the influence of the combiner to the overall system's overall performance. Simple combiners (Max combiner and single layer Perceptron combiner) are implemented. The detail experimental results are shown in Table 4.5.

	AdaBoost	S-AdaBoost (Max)	S-AdaBoost (Single Layer Perceptron)	S-AdaBoost (Multi Layer Perceptron)
Ringnorm	1.9 ± 0.4	**1.6** ± 0.1	1.6 ± 0.2	1.7 ± 0.2
Twonorm	3.0 ± 0.2	**2.7** ± 0.2	2.7 ± 0.3	**2.7** ± 0.2
Image	2.9 ± 0.9	2.9 ± 0.5	**2.6** ± 0.3	2.7 ± 0.5
Thyroid	4.5 ± 2.1	**4.0** ± 1.7	4.5 ± 2.4	4.3 ± 2.0
Splice	10.4 ± 1.1	9.6 ± 0.7	9.9 ± 1.0	**9.3** ± 0.8
Waveform	10.6 ± 1.3	10.0 ± 1.5	**9.6** ± 1.0	**9.6** ± 1.0
Banana	10.8 ± 0.8	10.6 ± 0.8	10.9 ± 0.4	**10.6** ± 0.5
Heart	20.8 ± 3.2	16.7 ± 3.3	19.5 ± 3.0	**15.9** ± 3.1
Titanic	23.1 ± 1.4	**22.0** ± 1.1	24.4 ± 1.6	22.2 ± 1.1
Diabetes	26.8 ± 2.0	24.7 ± 2.0	26.6 ± 2.0	**23.5** ± 1.6
German	27.5 ± 2.4	24.5 ± 2.2	26.7 ± 2.0	**23.8** ± 2.4
B. Cancer	30.8 ± 4.0	28.1 ± 4.2	30.7 ± 4.0	**26.1** ± 4.3
F. Solar	35.7 ± 1.6	33.5 ± 1.4	34.0 ± 1.8	**31.6** ± 1.8
Average Mean	16.1	14.7	15.7	14.1

Table 4.5: Comparison of the error rates among different combination methods on the benchmark databases

From the Table 4.5, it is shown that in the experiments, non-linear combiners (Max and Multi-Layer Perceptron based combiners) can achieve better performance than the linear one. The Inlier Classifier and the Outlier Classifier are of different structures and focuses, the non-linear ones can combine the results from the two classifiers more effectively.

Chapter Five

Applications: Using S-AdaBoost for Face Detection and Face Identification in the Complex Airport Environment

5.1 Introduction

The S-AdaBoost algorithm's effectiveness is demonstrated by the experimental results conducted on some benchmark databases through comparing with other leading outlier handling approaches in Chapter 4. To further demonstrate the effectives of the S-AdaBoost algorithm in the real world environment, two application systems, the face detection system FDAO and face identification system FISA are introduced in this chapter. FDAO system's performance is compared with the leading face detection approaches using the data obtained from both the complex airport environment and some popular face database repositories. The experimental results demonstrate the effectiveness of the S-AdaBoost algorithm on the face detection application in the real airport environment. Similarly, the FISA system is also based on S-AdaBoost algorithm and its performance is compared with the leading face identification approaches using the airport data and the FERET standard dataset. The results obtained are equally promising and convincing, which demonstrate that the S-AdaBoost algorithm is effective in handling the complex real airport environment in the face identification application.

5.2 The FDAO System

The FDAO system is an S-AdaBoost algorithm-based system designed for face detection system for the airport operators, which uses the surveillance cameras to scan crowds to extract potential face images and send the potential images for detecting whether it is a face image (as shown in Figure 5.1).

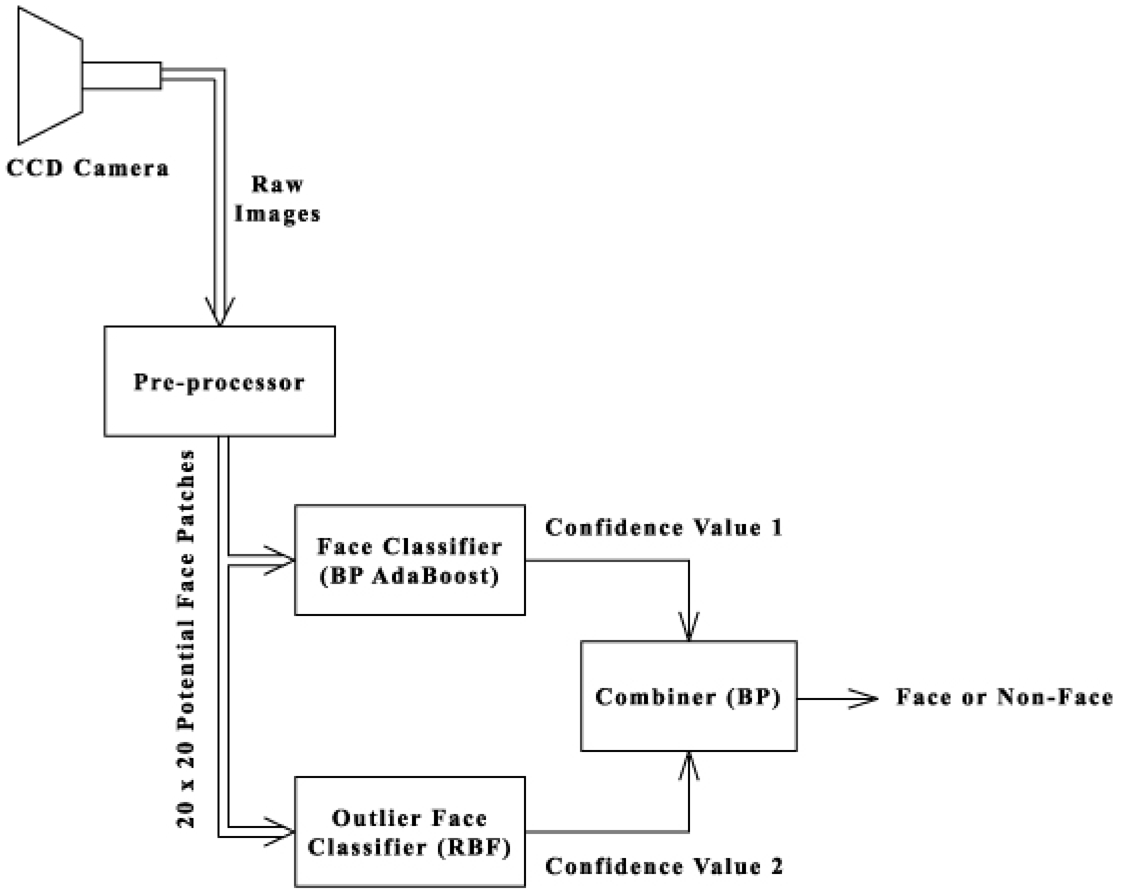

Figure 5.1 The FDAO system in use

In the experiment, the CCD cameras with a resolution of 320 X 256 pixels are installed in the airport to capture the raw images to be sent to the FDAO system. From Figure 5.1, it is observed that when the FDAO system is in use, the raw inputs from the CCD camera are fed to a pre-processor. The pre-processor acts as a filter to generate a series of potential face patches with 20 X 20 pixel resolution from the input image with the brightness normalized to a mean of 0 and a standard deviation of 1. Simple edge detection techniques are used to remove some of the obvious non-face patches. The pre-processor is designed in such a way to generate many more candidates than the real number of faces from the original images to avoid face images not being detected. The output 20 X 20 pixel potential face patches from the pre-processor are fed to both the face classifier **Đ(ŧ)** (whose base learner is implemented by an AdaBoost machine with the back-propagation neural network as the base weak learner) and the outlier classifier **O(x)** (implemented by a multi-layer RBF neural network) for classification. The reason of choosing the multi-layer feed-forward back-propagation neural network as the base weak learner for the face classifier is that Back Propagation Neural Network has good generalization capability. There are 400 input nodes in the input layer and 15 nodes in the hidden layer and one output node in the output layer; the 400 normalized gray level values act as the input to the input nodes, the conventional sigmoid function is used as the transfer function for the nodes, the adjacent layers are fully-connected. Hidden layer nodes enable the network to learn the complex image by progressively extracting more meaningful non-explicit features from the input patterns (as shown in Figure 5.2). As face patterns are highly non-linear [Sebastian and Lee, 2000], the non-linearity distributed form of the representation and the highly connected structure of the neural network base classifier suit the nature of the face detection problem. In the neural network, the input data is repeatedly presented to the neural network. After each round of input data presentation, the output of the neural network is compared with the desired output and an error is computed. This error is then fed back (back propagated) to the neural network and used to adjust the weights such that the error decreases with each iteration and the neural model gets closer and closer to the desired output.

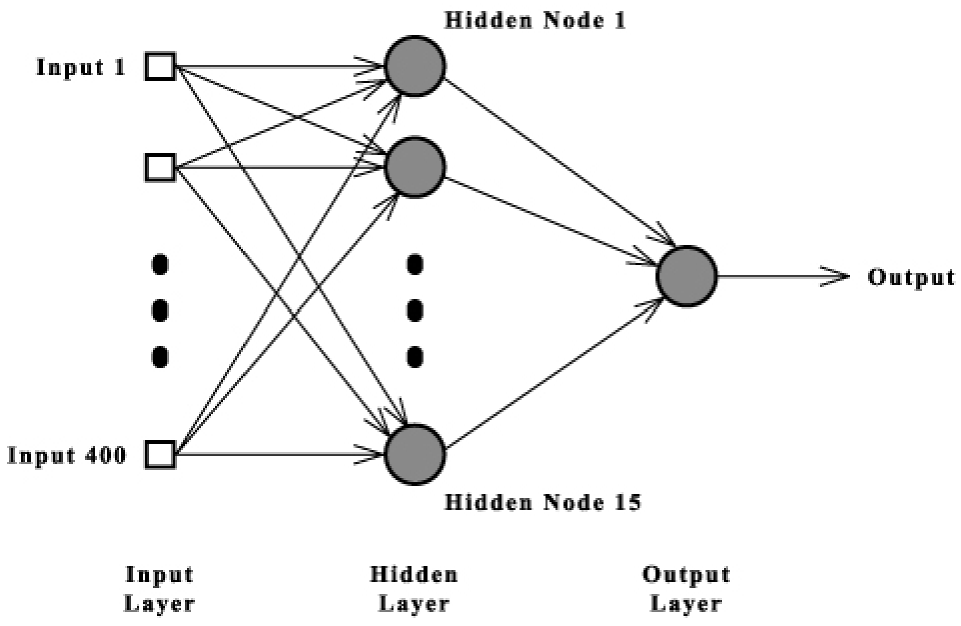

Figure 5.2 The back-propagation neural network base classifier in the FDAO system

A three-layer RBF neural network (as shown in Figure 5.3) is chosen to implement the outlier classifier $\mathbf{O}(\mathbf{x})$. The RBF (Radial Basis Function) neural network is chosen due to its good localization characteristic. Similar to the back propagation neural network base classifier, there are also 400 input nodes in the input layer in the RBF neural network and one output node in the output layer. The numbers of hidden nodes are dynamic and auto-growing depending on the diversity of the training patterns during the training stage. The radii of the hidden nodes in the RBF neural network are also chosen to be very small to enhance RBF network's good local clustering characteristic, which helps to isolate the noisy patterns $\mathbf{P}_{\mathbf{ns}}$ from the hard-to-classify patterns $\mathbf{P}_{\mathbf{hd}}$. The adjacent layers are also fully connected with each other. The RBF neural network uses a complete different methodology comparing with the back propagation neural network approach by projecting the input space to a higher dimensional space to achieve good localization capability. The hidden layer nodes use a set of "functions", which are called the radial basis functions to construct a new hidden feature space. As the face patterns are very difficult to separate, it is hoped that by using these hidden "functions", the face patterns could be localized in higher dimensional space by methods such as the linear separation, spherical separation or other separation schemes. Using the RBF neural network, along with the repeated presentation of the input patterns to the neural network, the input patterns are clustered to the centers formed in the hidden feature space; the number of hidden nodes is also automatically adjusted to optimize the radii of the clusters. Supervised learning method is used to learn the center locations in the hidden feature space as well as the weights of the output layer. Unlike the back propagation neural network used by the face classifier, the RBF neural network does not do error back-propagation. The RBF neural network is used as a local approximator whereas the back propagation neural network is used as a global approximator in the FDAO system.

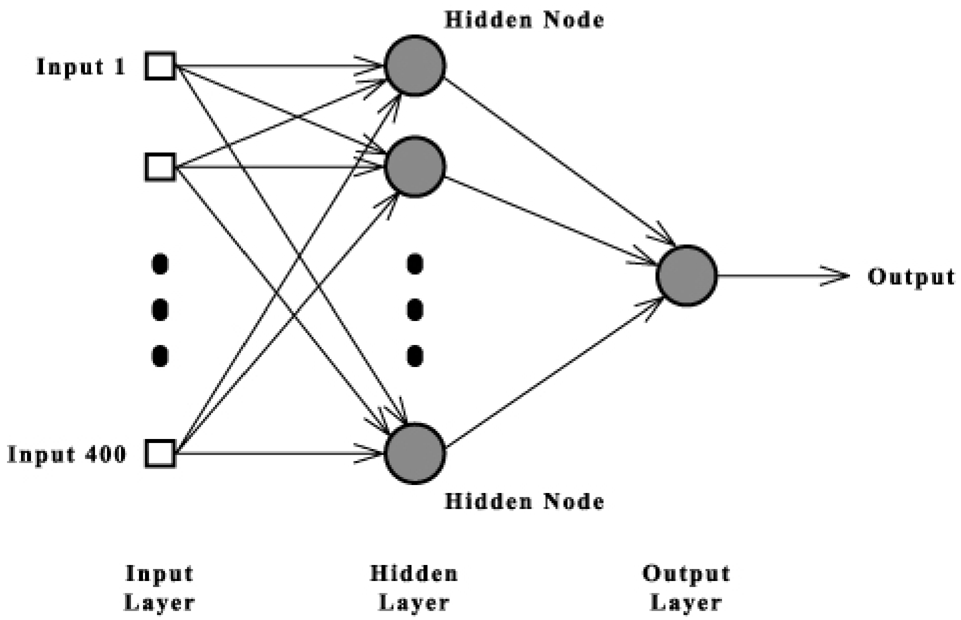

Figure 5.3 The radial basis function neural network outlier classifier in the FDAO system

Two confidence values act as the outputs of the face classifier $\mathbf{F_{od}(x)}$ and outlier classifier $\mathbf{O(x)}$ determining their estimates of whether the potential face image is indeed a face image. The confidence values act as the inputs to the combiner (implemented by a three layer back-propagation neural network as shown in Figure 5.4), which generates the final classification result determining whether the potential face patch of the input raw image is a face image or not. Multi-layer back propagation neural network is chosen to implement the combiner to make use of its good non-linear generalization capability. There are two input nodes in the input layer and three nodes in the hidden layer and one output node in the output layer. Conventional sigmoid function is used as the transfer function for the nodes; the adjacent layers are fully connected with each other as well. The hidden layer nodes enable the neural network to learn the complex relationship between the two confidence-values output by the two neural network classifiers. During training, for every new input, the output of the neural network is compared with the known judgment of whether the input is a face image and an error is thus calculated. This error calculated is then fed back to the back propagation neural network, which uses the error to adjust the weights of the combiner so that the system error rate decreases after every iteration and the neural network gets closer and closer to producing the correct final judgment. The final output is the classification result determining whether the input is a "face" or "non-face".

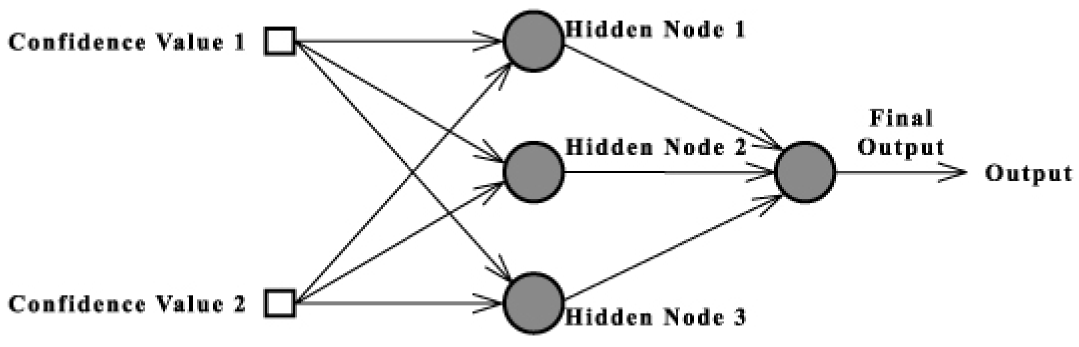

Figure 5.4 The back propagation neural network combiner in the FDAO system

An international airport environment is chosen as the complex environment to test the effectiveness of the FDAO system. There are tens of thousand passengers passing by this environment everyday, and potential face images are detected in complex backgrounds, which include different illumination, pose, occlusion, outlook and even make-up.

5.3 Training and pre-processing of the FDAO System

As mentioned in Chapter 2, one of the difficulties in face detection is that the face is often defined ambiguously. In FDAO system, following the definition by Yang [Yang Ming-Hsuan, Kriegman David, and Ahuja Narendra., 2002], face detection is defined as the detection of images with prominent facial features of mouth, nose and at least one eye. In the experiment, these 3 features are used to judge and analyze the results. It is found that most of the faces complying with this definition are frontal faces.

CCD cameras with a resolution of 320 X 256 pixels installed in the airport were used to collect the raw images to train the FDAO system. Out of all the images collected, 5000 images with one or multiple face images in the airport environment were selected for this experiment. The 5000 raw images were further divided into two separate datasets, one of the datasets contained 3000 raw images, and the other one contained the remaining 2000 raw images. More than 7000 face candidates were cropped by hand from the 3000 image dataset as the training set for the FDAO system; and the 2000 image dataset (some of them are shown in Figure 5.5) were chosen as the test set of the FDAO system. 5000 non-face images (including images of carts, luggage and pictures from some public image banks, etc.) were used (2500 images as the training set and the remaining 2500 images as test set) as non-face image dataset (some of them are shown in Figure 5.6). All the above training images were resized to 20 X 20 pixels and the brightness of the images were normalized to mean of zero and standard deviation of one before being sent for learning.

In the FDAO system, each image is preprocessed before being sent to the classifiers. To detect faces at arbitrary locations of a raw image, the patching is applied at various image locations. To detect face images larger than 20 x 20 pixels, the input image is repeatedly sampled, and the patching is applied at various scales. 20 x 20 pixel sub-images are then sent to the classifiers to generate the judgment result, which is the confidence value of a non-face (if the score is close to 0) or a face pattern (if the score is close to 1).

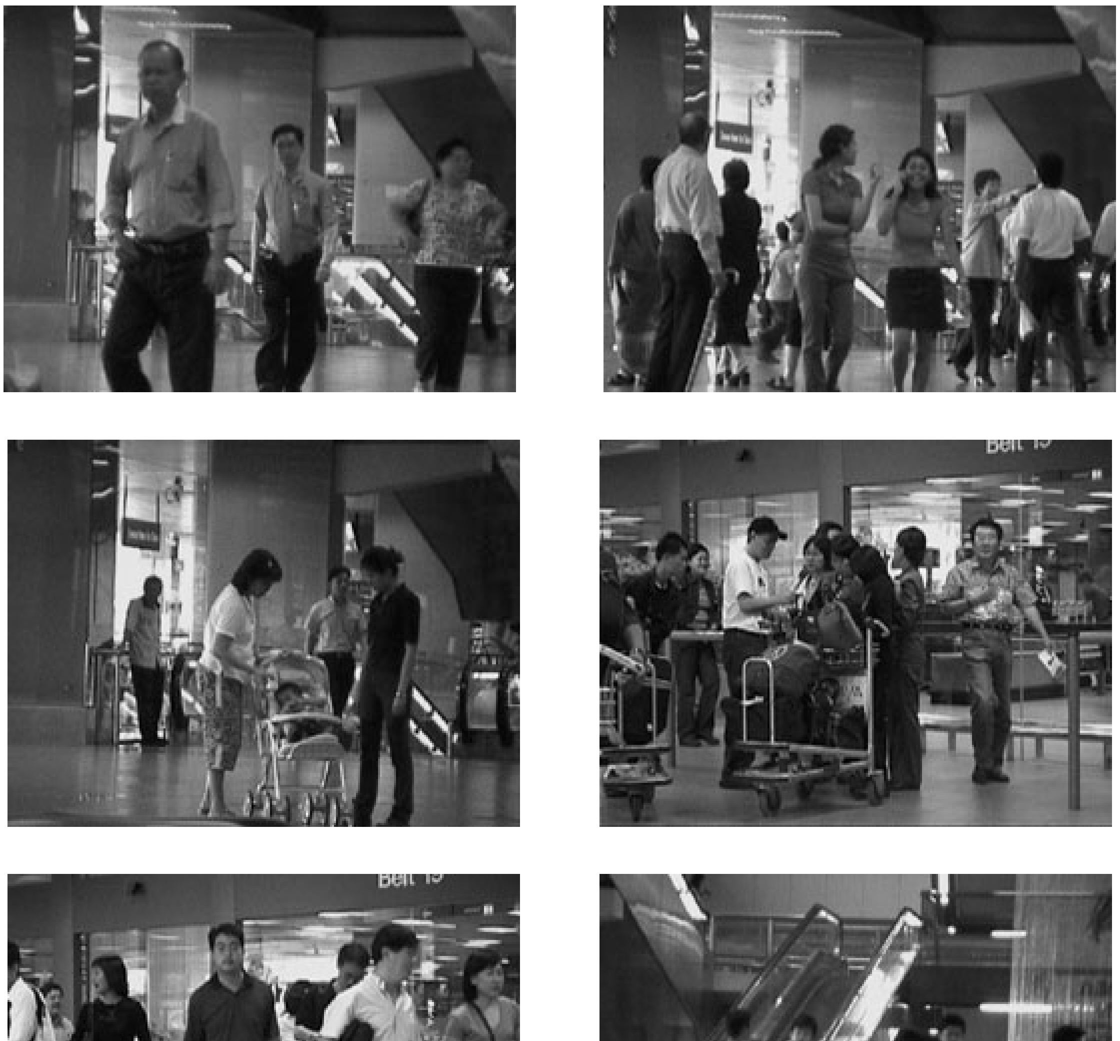

Figure 5.5 Some images containing faces used to test the FDAO system

Figure 5.6 Some non-face patterns used in the FDAO system

During training (as shown in Figure 5.7), the Divider Đ(ŧ) of the FDAO system is used to divide the whole potential face image stream into a normal face image sub-stream and an outlier sub-stream. The Divider Đ(ŧ) is constituted by a dividing network and a gating mechanism (as shown in Figure 5.8), which jointly generate different training sets for the following two classifiers: the back-propagation neural network based face classifier $\mathbf{Ƒ_{od}(x)}$ and the multi-layer RBF neural network based outlier classifier **O**(x). Similar to the face classifier $\mathbf{Ƒ_{od}(x)}$**,** multi-layer back-propagation neural network is chosen to be the base classifier to implement the Divider Đ(ŧ) dividing network. The Divider Đ(ŧ) dividing network is implemented by an AdaBoost machine with 400 input nodes in the input layer and 15 nodes in the hidden layer and one output node in the output layer (the same as the AdaBoost face classifier $\mathbf{Ƒ_{od}(x)}$); the 400 normalized gray level values act as the input to the Divider Đ(ŧ) input nodes, the conventional sigmoid function is used as the transfer function for the Divider Đ(ŧ) nodes; adjacent layers in the Divider Đ(ŧ) dividing network are fully-connected. Hidden layer nodes in the Divider Đ(ŧ) dividing network enable the system to learn the complex input images by extracting progressively more meaningful non-explicit features (the same as the face classifier $\mathbf{Ƒ_{od}(x)}$). The output of the Divider Đ(ŧ) dividing network is fed to a gating mechanism (as shown in Figure 5.8, if the judgment result shows that the input image is an "Inlier", the image is sent to $\mathbf{Ƒ_{od}(x)}$, otherwise **O**(x).) together with the 20 X 20 pixel potential face images to decide whether to direct the 20 X 20 pixel potential face images to the back-propagation neural network based face classifier $\mathbf{Ƒ_{od}(x)}$ or the multi-layer RBF neural network outlier classifier **O**(x).

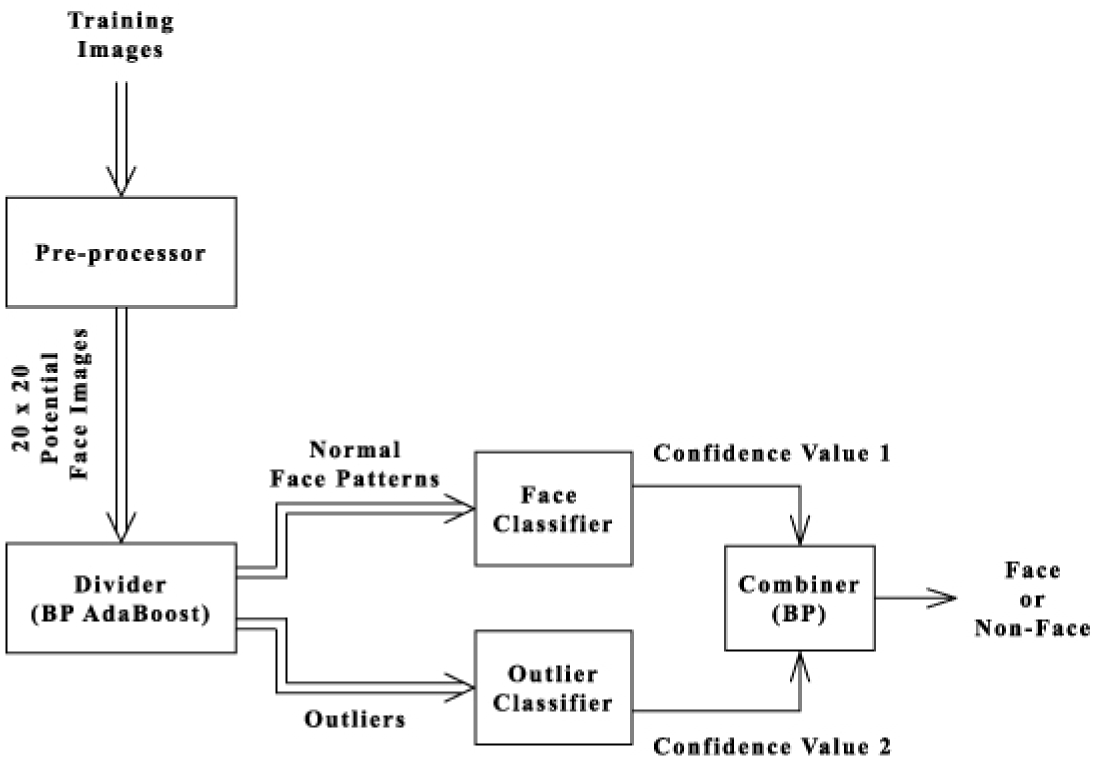

Figure 5.7 Training the FDAO system

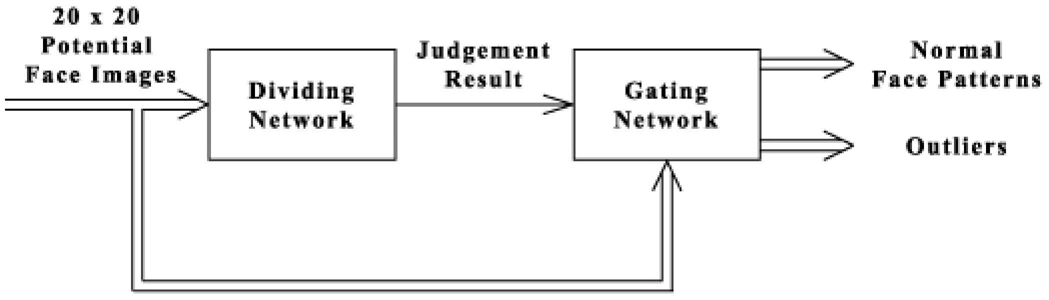

Figure 5.8 The dividing network and the gating mechanism of the Divider Đ(t) in the FDAO system

In the experiment, the 3000 training raw images together with the 2500 non-face images were sent to the pre-processor of the FDAO system to generate a series of potential 20 X 20 pixel face patches, which subsequently act as the training input to the Divider (ŧ) during training. The training inputs to the back-propagation neural network based face classifier **F_{od}(x)** and the multi-layer RBF neural network outlier classifier **O(x)** were taken from the divided stream output by the Divider (ŧ). The confidence values output by the face classifier $\mathbf{F_{od}(x)}$ and outlier classifier $\mathbf{O(x)}$ act as the training input to the Combiner $\hat{\mathbf{C}}$ of the FDAO system.

5.4 Face Detection Experimental Results

In this section, the FDAO system experimental results in the complex airport environment are presented. Other leading face detection algorithms are also discussed, implemented using the same airport dataset used in the FDAO system experiment. Results obtained from these methods are compared and analyzed. To further prove the S-AdaBoost algorithm's effectiveness in the face detection application, experiments are also conducted on some benchmark face detection databases to compare with the results obtained from other leading approaches on these benchmark datasets. Some testing patterns are also sent to CMU (Carnegie Mellon University) face group's web-based on-line face detection program for obtaining comparison results.

5.5 The Test Results from the FDAO System

To obtain the optimal threshold value $\mathbf{t}$, different sets of test results are generated by the FDAO system through choosing different threshold value $\mathbf{t}$ for the Divide ($\mathbf{t}$). Different sets of the corresponding face classifier $\mathbf{F_{od}(x)}$ and outlier classifier $\mathbf{O(x)}$ are also trained based on the different output of the Divider ($\mathbf{t}$) for different threshold value $\mathbf{t}$. To measure the effectiveness of the FDAO system, two error rates are measured, namely the false positive rate as well as the detection error rate δ, which is defined as:

$$\text{Detection Error Rate } \delta = \frac{(N_1 + N_2)}{N} \qquad (5.3.1.1)$$

Where N1 stands for the number of the face images wrongly classified as non-face images;

N2 stands for the number of the non-face images wrongly classified as face images; and

N stands for the total number of faces in the FDAO test set.

The corresponding false positive rates and the detection error rates for different threshold value **t** are plotted in Figure 5.9.

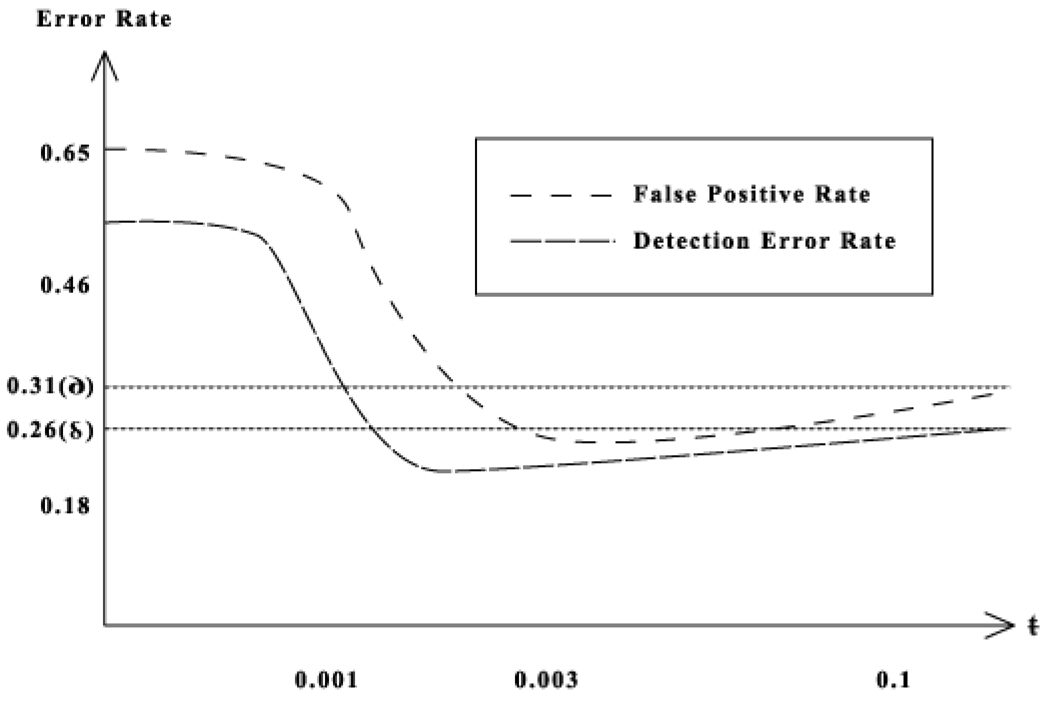

Figure 5.9 Error rates of the FDAO system

In Figure 5.9, the Y-axis denotes the error rate; the X-axis (not drawn proportionally) denotes the value of the threshold value **t**. It is found that the error rates of the FDAO system decrease slowly when the threshold value **t** gradually increases from 0 (when all patterns are treated as outliers) upward, the error rates of the FDAO system drop faster and faster with the increment of the threshold value **t** before becoming stable for a while. The error rates slowly climb up afterwards; in the end, when the threshold value $\mathbf{t}$ approaches 1**,** the false positive rate reaches ∂ and the detection error rate of the system reaches δ. To explain this observation, the input patterns in the outlier set in the pattern space $\mathbf{\hat{S}_{ol}}$ are examined, it is found that when the threshold value $\mathbf{t}$ is small (near value 0), most of the patterns in the input space $\mathbf{\hat{S}}$ are in the outlier pattern space $\mathbf{\hat{S}_{ol}}$, and the system's generalization performance is affected, which results in the high error rates. Along with the increment of the threshold value $\mathbf{t}$, there are more and more normal patterns $\mathbf{P_{no}}$ and special patterns $\mathbf{P_{sp}}$ going into the ordinary pattern space $\mathbf{\hat{S}_{od}}$; more and more numbers of the genuine clustering of the hard-to-classify patterns $\mathbf{P_{hd}}$ are detected in the outlier space $\mathbf{\hat{S}_{ol}}$, the error rates of the system go down faster and faster before reaching an optimal range (in this case from value 0.001 to value 0.005). When the threshold value $\mathbf{t}$ increases further, the hard-to-classify patterns $\mathbf{P_{hd}}$ and the noisy patterns $\mathbf{P_{ns}}$ start to enter the ordinary space $\mathbf{\hat{S}_{od}}$, the face classifier $\mathbf{F_{od}(x)}$ tries progressively harder to adopt these input patterns, which results in the slow increase of the error rates of the FDAO system as well. The false positive rate reaches ∂ and the Detection Error Rate reaches δ when all the input patterns go to the ordinary space $\mathbf{\hat{S}_{od}}$. It is noted that most of the time (when the threshold value $\mathbf{t}$ exceeds value 0.001), S-AdaBoost achieves better false positive rate and the detection error rate than those of the AdaBoost machine (0.26).

The number of training patterns M

= 7000 face (in the 3000 face images) + 2500 non-face images

= 9500

The error detection rate of the AdaBoost Machine:

$\delta = 0.26$;

If the threshold value is set to be equal to $1/(M \text{ X } \delta^2)$, then

The threshold value $\mathbf{t}$

$= 1/(M \times \delta^2)$

$= 1 / (9500 \times 0.26^2)$

$= 0.0015$

It is noted that this threshold value t falls into the optimal range (from value 0.001 to 0.005).

This formula gives a guideline in setting the threshold value; a more accurate value can be decided by experiment.

5.6 Testing Results of the Other Leading Face Detection Algorithms in the Complex Airport Environment

To test the effectiveness of the S-AdaBoost algorithm on the face detection application in the complex airport environment, the performance of the FDAO system (when the threshold value t is set to $1/(M \times \delta^2)$, where M is the number of all the training patterns and the parameter δ is the detection error rate of the AdaBoost machine.) and other leading approaches are compared. The Rowley, Baluja and Kanade's [1998] Neural Network approach, the Viola and Jones's [2001] Asymmetric AdaBoost Cascading approach, and the Osuma, Freud and Girosi's [1997] Support Vector Machine approach are implemented. To make the approaches more comparable, the Detection Error Rate δ of the four algorithms are compared in the test.

In the Rowley, Bajula and Kanade's Neural Network approach, each face is preprocessed before being fed to an ensemble of neural networks, which is used to detect the face patterns. A decision making structure (similar to the structure of the Combiner in the S-AdaBoost machine) is used to render the final decision from the multiple detection results) follows the neural network ensemble to make the final judgment. After the pre-processing, a 20 x 20 pixel sub-image (the same size as that of S-AdaBoost sub-images) are sent to the neural network ensemble and the scores ranging from -1 to 1 are generated, which is the confidence value of a non-face (if the score is close to -1) or a face pattern (if the score is close to 1) (in the S-AdaBoost machine, the confidence values range from 0 to 1). To detect faces at arbitrary locations of a raw image, the patching is applied at all image locations. To detect face images larger than 20 x 20 pixels, the input image is repeatedly sampled, and the patching is applied at each possible scale (the same method is also used by FADO). More than 1000 face patterns of various sizes, orientations, positions, and intensities are used to train the neural network ensemble. For each training image, the eyes, tips of the nose, corners, and centers of the mouth are manually cropped and labeled. The normalization process is conducted to normalize the face images to the same scale, orientation, and position. The decision-making structure takes the output from the individual neural networks and detects duplicated detection among the individual neural networks. Simple arbitration schemes such as Logic AND and Logic OR operators as well as voting, instead of the complex ones used by some researchers (such as operators used by [Sung and Poggio, 1998]) are used to make the final judgment. Satisfying results are obtained in their experiments. (In the S-AdaBoost machine, due to the different structure of the two neural network classifiers, a complex combination mechanism is used instead). The system has also been extended to detect possible face rotation using a special router network, which processes each input window to determine the possible face orientation and then rotate the window to a canonical orientation, before feeding the potential face images to the neural network ensemble for detection.

Rooted in the statistical learning theory, Osuma, Freud and Girosi's SVM (Support Vector Machines) based face detection algorithm also receives much attention. Different from the ERM (Empirical Risk Minimization) principle used by the conventional Neural Network based face classifiers (such as the Rowley, Bajula and Kanade's approach), the Support Vector Machine based face detection approaches apply the SRM (Structural Risk Minimization) principle and use the RBF (Radial Basis Function) based face classifier or face polynomial function based classifiers to reduce the structural risk of the system. Instead of minimizing the training error as practiced in the ERM based approaches, the upper bound of the expected generalization error is minimized in the SRM based approaches (the Regularized AdaBoost approach discussed before also falls into the SRM category). A small subset of the training vectors, which are called the support vectors, are chosen to form an SVM (Support Vector Machine) classifier, which is a linear classifier (if the non-linear kernel is sued, the classifier can be expanded to non-linear) chosen to minimize the expected classification error of the test patterns. A linearly constrained quadratic programming problem needs to be solved to estimate the optimal hyper-plane formed by the classifier. The Osuma, Freud and Girosi's SVM based face detection approach successfully handles a large-scale face detection problem and achieved good experimental results on the two test sets consisting of 10,000,000 test patterns of 19 x 19 pixel resolution (in the FDAO system, the patterns are normalized to 20 X 20 pixel resolution with mean of zero and standard deviation of one).

Viola and Jones's Asymmetric AdaBoost Cascading approach is one of the most rapid approaches able to conduct real-time face detection in a normal 700M Hz Intel Pentium III based notebook and process the 15 fps (frame per second) 384 X 288 pixel gray level input images. Viola and Jones claims that the Asymmetric AdaBoost

Cascading approach has made three key contributions to face detection research. The first contribution is the introduction of a new image representation scheme called “Integral Image", which enables the features used by the Asymmetric AdaBoost Cascading detector to be computed and evaluated rapidly. The second contribution is the introduction of a new AdaBoost based learning algorithm, which is able to make use of just a small number of critical visual features to build the efficient classifiers. In order to achieve efficient system performance, only a small percentage of the critical features are selected through a modified AdaBoost procedure, which constrains the weak base classifier so as to make each weak classifier only focus on one feature. In this regard, every iteration of the AdaBoost algorithm can be considered as a new feature selection process. The reason of choosing the AdaBoost algorithm to build the face detection classifier is because of the AdaBoost algorithm’s effectiveness and the good generalization capability. The third contribution claimed by Viola and Jones is the introduction of the combining cascading classifiers, which is able to discard the image background and focus on identifying the promising objects to speed up the classification process. A set of face detection experiments have been conducted by Viola and Jones to show the effectiveness of the Asymmetric AdaBoost Cascading approach.

After implementing the above three leading face detection systems, the same training set (the face and non-face datasets) and the testing set (the face and non-face datasets) used in the FDAO system are used to train and test these three systems to compare the effectiveness of different approaches in the real complex airport environment. Again, the 10 cross validation method is used in the experiment. The testing results obtained from the leading face detection approaches are listed in Table 5.1.

	Rowley	Viola	SVM (RBF Kernel)	S-AdaBoost
Average Detection Error Rate	29.4%	27.1%	27.7%	25.5%
Standard deviation	± 3.2%	± 2.9%	± 3.0%	± 3.5%

Table 5.1: Comparison of error rates of the different face detection approaches

From Table 5.1, it can be found that the S-AdaBoost approach achieves the best experimental result compared with the other three leading approaches using the training and testing datasets obtained in the complex airport environment. Even though other approaches achieve good results in their respective face detection databases, they seem to suffer more compared with the FDAO system when the datasets are "noisy" with a large amount of outliers. Utilizing AdaBoost to implement the dividing mechanism and utilizing a dedicated RBF outlier classifier in the S-AdaBoost based FDAO system is proven effective in this round of test in the complex airport environment.

Further comparison between the results in Table 4.2 and those in Table 5.1 shows that S-AdaBoost out-performs other methods in the latter more than in the former. Analysis shows that this might be due to the fact that the data collected in FDAO is more "raw" and "noisy" than the data collected in the benchmark databases used in Table 4.2.

5.7 Comparison of the Leading Face Detection Approaches on the Standard Face Detection Databases

In order to conduct a fair comparison between the S-AdaBoost algorithm-based system and the other leading approaches. All the algorithms should be trained and tested on the same datasets. In the previous section, the datasets collected from the complex airport environment are used to conduct the training and testing. In this section, the current popular face databases and the benchmark face detection benchmark databases are studied and explored to find the possibility to use one of, or a combination of a few of, them to do another round of tests to the face detection algorithms.

The researchers in MIT (Massachusetts Institute of Technology) have produced at least two standard face detection databases. The MIT media lab database (ftp://whitechapel.media.mit.edu/pub/images/) consists of the face images from 16 people, 27 images are collected for each person under different conditions of illumination, scale and head orientation. Another popular face image database from MIT is the MIT CBCL (the Center for Biological and Computational Learning) face dataset (http://www.ai.mit.edu/projects/cbcl/software-datasets/FaceData2.html). The MIT CBCL database is further divided into a training set and a testing set, the training set consists of 6,977 cropped images (2,429 faces and 4,548 non-faces), and the testing set consists of 24,045 images (472 faces and 23,573 non-faces). All the images are normalized to 19 X 19 pixel resolution with values between 0 and 1; a tag follows the data indicating whether it is a face image (-1 for a non-face image and 1 for a face image).

The FERET database is another database used for face detection and identification applications, the FERET database will be discussed further in the face identification application system in the next few sections. The UMIST (University of Manchester Institute of Science and Technology) face database (http://images.ee.umist.ac.uk/danny/database.html) consists of 564 images of 20 people. Images are of 220 X 200 pixel 8 bit gray level capturing people of different race, gender and appearance. The images are of different poses (from profile to frontal views) as well. The AT&T Cambridge lab face database (formally known as the ORL database of faces) (http://www.uk.research.att.com/facedatabase.html) contains images of 40 subjects. There are ten different images for each of the 40 distinct subjects. For some subjects, the images are taken at different time, lighting, facial expressions (open / closed eyes, smiling / not smiling) and facial details (glasses / no glasses). All the images are taken against a dark homogeneous background with the subjects in an upright, frontal position (with tolerance of some side movement). The size of each image is 92 X 112 pixels, with 256 gray levels per pixel.

The M2VTS (Multi Modal Verification for Tele-services and Security applications*)* face database (http://www.tele.ucl.ac.be/PROJECTS/M2VTS/ m2fdb.html) is a multi-modal database containing various image sequences. It is made up of 37 different faces with 5 shots per person. The shots are taken at one-week intervals or when drastic face changes occur. During each shot, people are asked to rotate their heads from 0 degree to -90 degrees, again to 0 degree, then to +90 degree and back to 0 degree. Also, they are asked to rotate their heads once again without glasses if they wear any. The *motion* sequence and the *glasses off* motion sequence (if any) are captured. The two sequences can provide information about the 3-D face features thanks to the motion. They may also be used to implement or compare with other techniques like identification from 2-D facial pictures, profile view or multiple views. These shots mainly differ from the others because of face variations (head tilted, eyes closed, different hairstyle, presence of a hat/scarf and etc.), or shot imperfections (poor focus, different zoom factor, etc.). The resolution for the database images is 286 X 350 pixels.

Purdue face database (http://rvl1.ecn.purdue.edu/~aleix/aleix_face_DB.html) is created by Aleix Martinez and Robert Benavente. It contains over 4,000 color images corresponding to 126 people's faces (70 men and 56 women). Images feature frontal view faces with different facial expressions, illumination conditions, and occlusions (wearing sun glasses or scarf). The pictures are taken under strictly controlled conditions. No restrictions on wear (clothes, glasses, etc.), make-up, hair style, etc. are imposed to the participants. Each person participates in two sessions; the shots are taken in two-week intervals and the same pictures are taken in both sessions. Images are of 768 by 576 pixels and of 24-bit color. The 3,276 face images are with different facial expressions and occlusions under different illuminations.

Some benchmark databases have also been developed in the past. CMU frontal face test set [http://vasc.ri.cmu.edu/IUS/eyes_usr17/har/har1/usr0/har/faces/test/] has been a popular benchmark dataset to evaluate the performance of a face detection system to detect frontal faces in grayscale images. Three test sets are included in this collections, which are referred as Test Set A, Test Set B, Test Set C and Rotated Test Set. Test Set B is provided by Kah-Kay Sung and Tomaso Poggio at the AI Lab at MIT, and Test Set A, Test Set C and Rotated Test Set are collected by Henry A. Rowley, Shumeet Baluja, and Takeo Kanade in CMU. All the files are of gray level and GIF format. A new database called the CMU Pose, Illumination, and Expression (PIE) has also been developed in CMU [Sim, Baker, and Bsat, 2003]. It contains 41,368 images of 68 people. The images of each person are taken under 13 different poses, 43 different illumination conditions, and with 4 different expressions.

In our experiment, the training set was built from the MIT CBCL Face Dataset (1000 face images and 1000 non-face images were abstracted from MIT CBCL dataset), and Purdue Databases (1000 face images from Purdue dataset); the CMU frontal databases (contains 130 images of 507 frontal face images) as the test set. MIT CBCL dataset was re-sized from the resolution of 19X19 to 20X20, and Purdue data was re-sized to gray-level 20 X 20 from 768 X 576 color pictures. 5 approaches (Rowley's approach, Viola's approach, the SVM approach (RBF Kernel), the AdaBoost approach and the S-AdaBoost approach) were implemented (We thank some researchers for providing some of their source codes used here). The test results are listed in Table 5.2:

Approaches	δ Mean%	δ Standard deviation %
Rowley	5.5	0.6
Viola	4.8	0.5
SVM	4.1	0.6
AdaBoost	6.2	0.6
S-AdaBoost	3.9	0.7

Table 5.2: Comparison of error rates among various methods on CMU-MIT databases.

Table 5.2 shows that all the algorithms are stable algorithms with generally low and similar standard deviation. The S-AdaBoost based approach achieves the best results in the experiment. All the other three algorithms perform better than pure AdaBoost based approach; the SVM based approach achieves the second best result (very near to the performance of the S-AdaBoost based approach). These experimental results further demonstrate the effectiveness of the S-AdaBoost algorithm. In the experiment, face alignment and "integral face" (as used by Viola) are not used. Facial alignment has been tested but not reported in the thesis; as due to the highly complex environment, facial alignment does not work well in our scenario. The "integral face" approach is mainly for fast detection, as our system can achieve real time detection using a 60fps CCD camera connected to a PC, the "integral face" approach is not used in our system.

5.8 Comparison with the CMU on-line Face Detection Program

As a further comparison, 50 testing images (including the 8 images shown in Figure 5.5) are sent to the CMU on-line face detection test program [http://www.vasc.ri.cmu.edu/cgi-bin/demos/findface.cgi] for analysis [http://vasc.ri.cmu.edu/demos/faceindex/images Submission 1-13 on October 19, 2002 and Submission 4-40 on Oct 18, 2002]. The detection error rate obtained from the 50 testing image set is 55% and the number of false face images detected is 23 (some sample results are shown in the Fig 5.10). Again, as a comparison, 50 testing face images were sent to the FDAO system; the detection error rate obtained from the FDAO system on the 50 testing image set is 20% and the number of the false face images detected is 8 (some sample results are shown in the Fig 5.11).

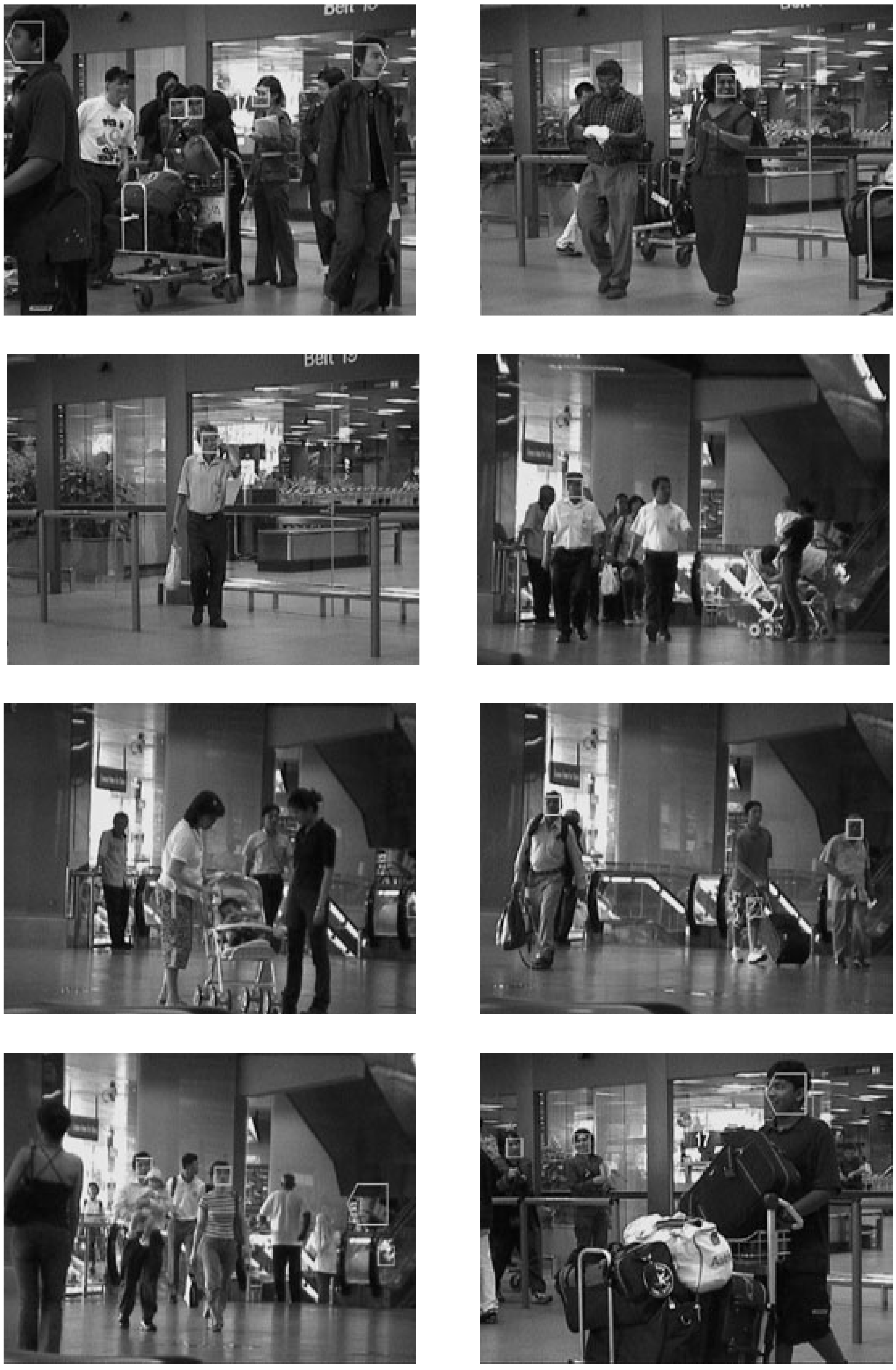
Figure 5.10 Sample results obtained from the CMU on-line face detection program on some face images

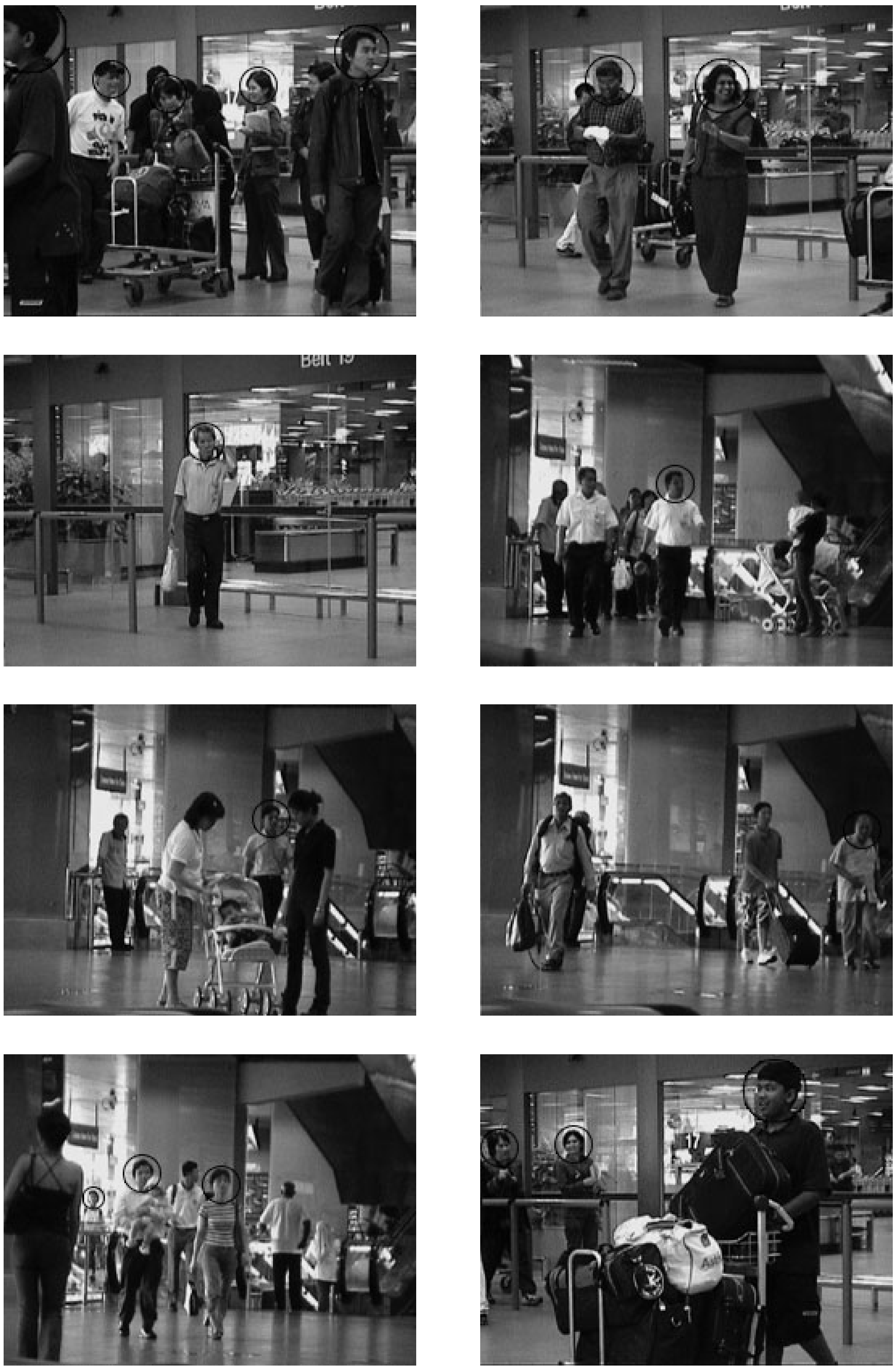
Figure 5.11 Sample results obtained from the FDAO system on some face images

For easy description, the sample figures are named following the convention that the top-left sample image is called sample image number one, the top-right image is called sample image number two, the bottom right image is the last image, the rest are named accordingly.

It is noticed that there is one face being correctly detected, and one false face being detected in sample image three in Figure 5.10; there are one face being correctly detected, and no false face being detected in sample image three in Figure 5.11. Table 5.3 lists the comparison of the detection results of the two algorithms on these 8 face image samples. From Table 5.3, it is shown that the FADO system performs better than the CMU on-line program especially in the reducing the number of false detections among the testing samples.

Image Number	No. of correct face detections in Figure 5.10	No. of correct face detections in Figure 5.11	No. of false face detections in Figure 5.10	No. of false face detections in Figure 5.11
1	5	5	0	0
2	1	2	0	0
3	1	1	0	0
4	1	1	1	0
5	0	1	1	0
6	2	1	1	0
7	2	2	2	1
8	3	3	0	0

Table 5.3: The detection results of the CMU on-line program and the FDAO system on the 8 samples

Later, 63 testing images (including the 8 images shown in Figure 5.6) were sent to the CMU on-line face detection test program for analysis as well [http://vasc.ri.cmu.edu/demos/faceindex/images Submission 1 - 63 on Friday April 18, 2003]. The total number of images with false face detection is 11 and the total number of false face detected is 21 (some false detection sample results are shown in the Fig 5.12). Again, as a comparison, 63 testing non-face images were sent to the FDAO system. The total number of images with false face detection is 5 and the total number of false face detected is 7 (some false detection sample results are shown in the Fig 5.13).

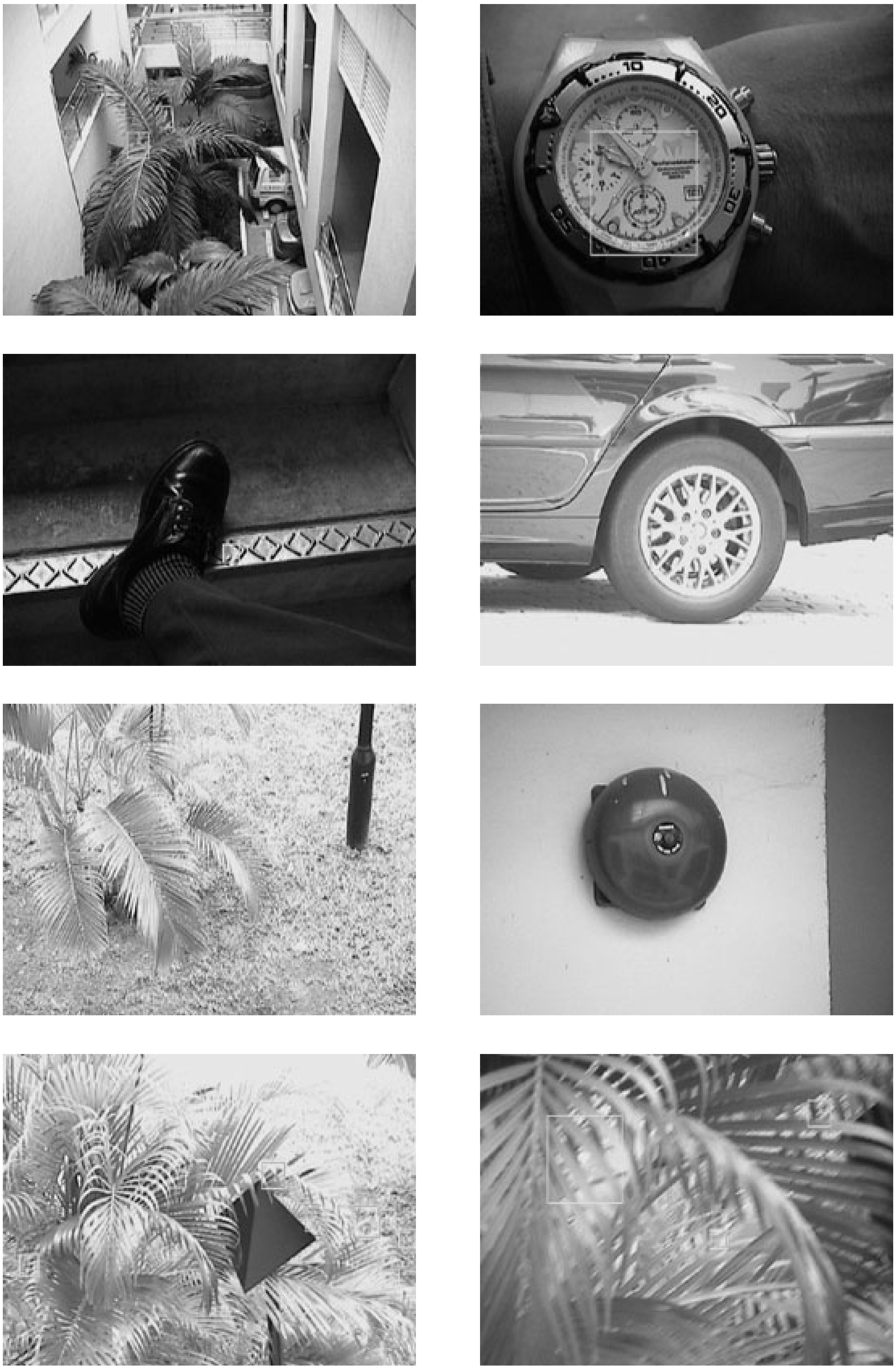

Figure 5.12 Sample results obtained from the CMU on-line face detection program on some non-face images

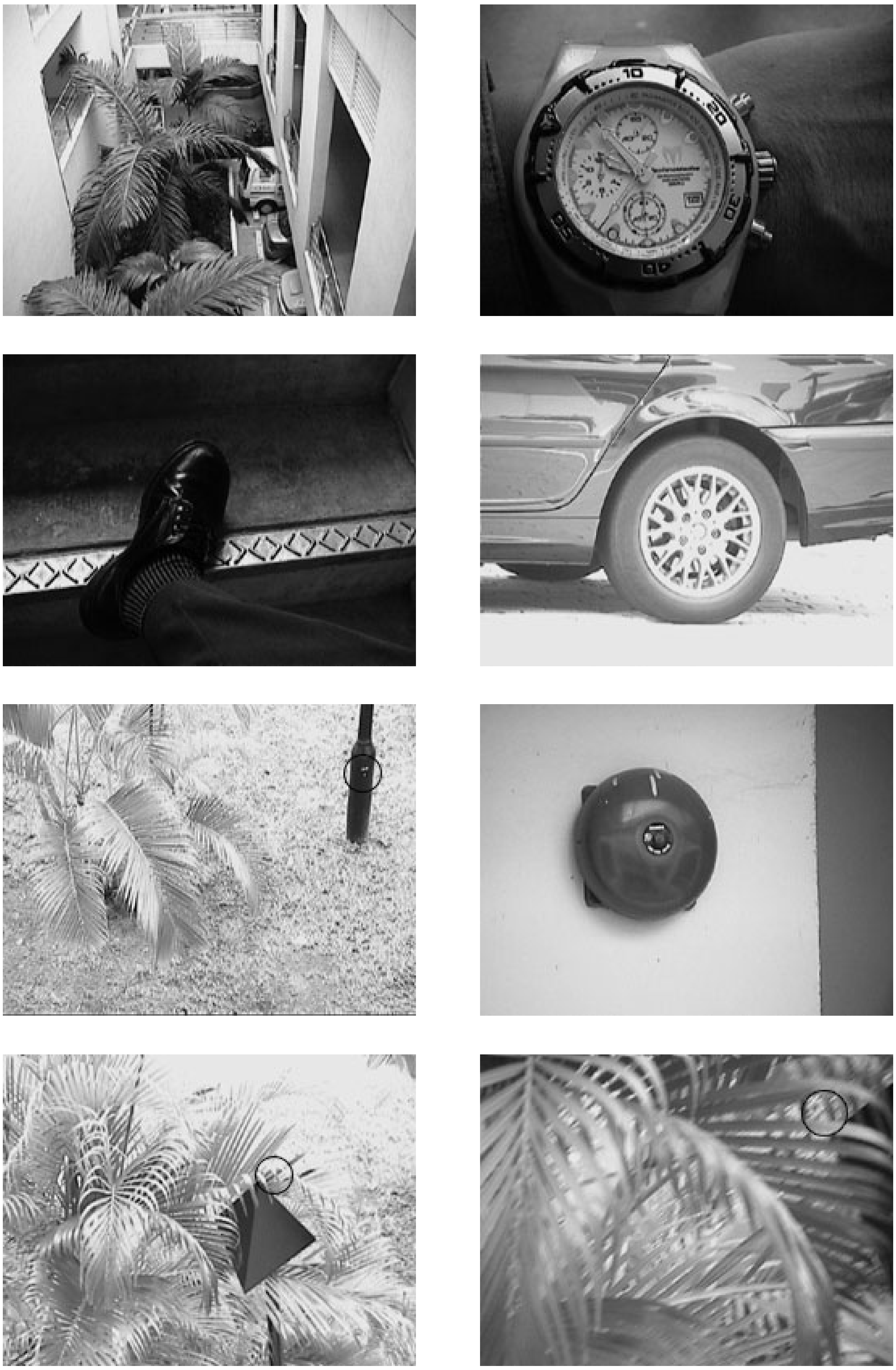

Figure 5.13 Sample results obtained from the FDAO system on some non-face images

Table 5.4 lists the comparison of the detection results of the two algorithms on these 8 non-face image samples. From Table 5.4, it is shown very clearly that the FADO system's performance in detecting non-face images is much better that that of the CMU on-line program.

Image Number	No. of false face detections in Figure 5.12	No. of false face detections in Figure 5.13
1	1	0
2	1	0
3	1	0
4	0	0
5	0	1
6	0	0
7	3	1
8	2	1

Table 5.4: The detection results of the CMU on-line program and the FDAO system on the 8 non-face samples

5.9 Face Identification using the S-AdaBoost Algorithm

To demonstrate the effectiveness of S-AdaBoost, the face identification application system FISA (Face Identification System for Airports) [Liu and Loe, 2003b] using the S-AdaBoost algorithm in the complex airport environment will be introduced. Similar to the comparative method used in the previous sections, the experiment obtained from the FISA system is compared with the results of the other leading approaches obtained from the complex airport environment as well as some benchmark datasets.

5.9.1 Face Identification and the FISA System

Different from the face detection systems introduced in the previous sections, the face identification system takes an image as its input and reports back the identities of the people in the image after searching through a database containing all the known candidates. In the airport environment, due to the large number of moving passengers, a non-intrusive face identification system is in urgent demand to scan the crowd to look for the potential wanted terrorists or the person of interest. Integrating with other technologies such as the smart card technology, the face identification system can be used for immigration control and custom clearance in the airport environment as well.

Even though it has been demonstrated in the previous sections that the S-AdaBoost based approach is effective in the face detection application in the complex airport environment, it is still unclear whether the S-AdaBoost based approach can achieve the same effectiveness in the face identification application. Some research shows that a face detection system makes use of the low-frequency features of the images, whereas the face identification system focuses on the high-frequency features of the images [Sergent, 1986]. Further research shows that if the distinct face features are stored in the memory based network, the face identification system using this network will result in faster and better identification; whereas the face detection using the same network will lead to much slower and more inaccurate results. In Addition, the complex environment (such as the variations of illumination, pose and others) is also challenging the design of the face identification systems.

The S-AdaBoost algorithm is based on the "divide and conquer" strategy. According to some researchers [Biederman and Kalacsai, 1998], compared with the face detection system, there are at least seven features that should be considered when designing the face identification system. They are the differences in the configural effects; special expertise; whether difference between faces can be verbalized; the sensitivity to contrast polarity and illumination direction; the difference in metric variation; the extent of rotation of the face images; and the inverted image effect.

The FISA system, which stands for the Face Identification System for Airports is an S-AdaBoost algorithm-based system. It is used to spot the identities of 40 people from a complex airport environment with thousands of people passing the airport everyday. Images with complex background are sent to the system and the identities of the people of interest will be highlighted by the system automatically. Figure 5.14 shows a typical scenario of the system.

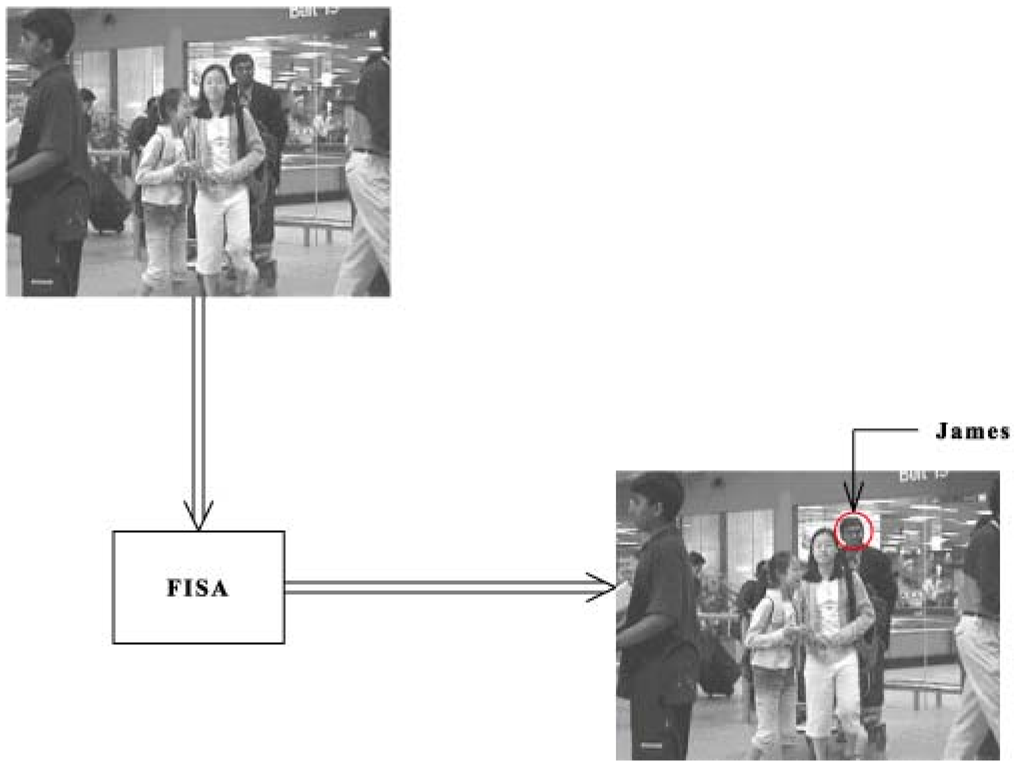

Figure 5.14 A typical scenario in the FISA System

The face identification system FISA (as shown in Figure 5.15) includes a Face Detection Module (FDM) and an Identity Detection Module (IDM). The face detection system FDAO discussed in the previous chapter can be used to implement the FDM in the FISA system. This means that if there are face images detected by the FDAO system, the corresponding 20 X 20 pixel potential face images will be sent to the IDM module for identity check. As the FDAO system only achieves about 75% face detection rate in the experiment conducted in the previous sections, some adjustment in the FDAO Combiner is done to ensure more potential face images are passed to the identity detection module IDM for identity check in the FISA system. The focus of the following discussion is the design of the IDM module in the FISA system.

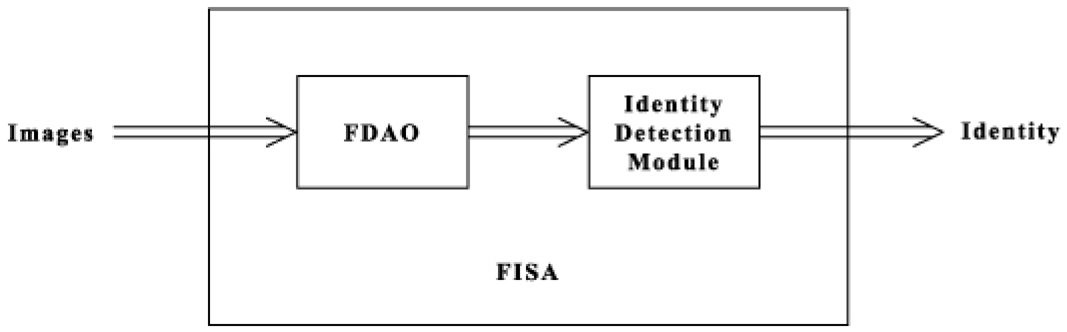

Figure 5.15 The FISA system

In the training phase, the IDM module of the FISA system is implemented by a structure (as shown in Figure 5.16), which is very similar to that of the FDAO system in training. The 20 X 20 pixel detected potential face images from the FDAO system (with brightness normalized with mean of zero and standard deviation of one) are fed to the IDM module for identity check. The Divider in the IDM module divides the potential face images into inliers and outliers.

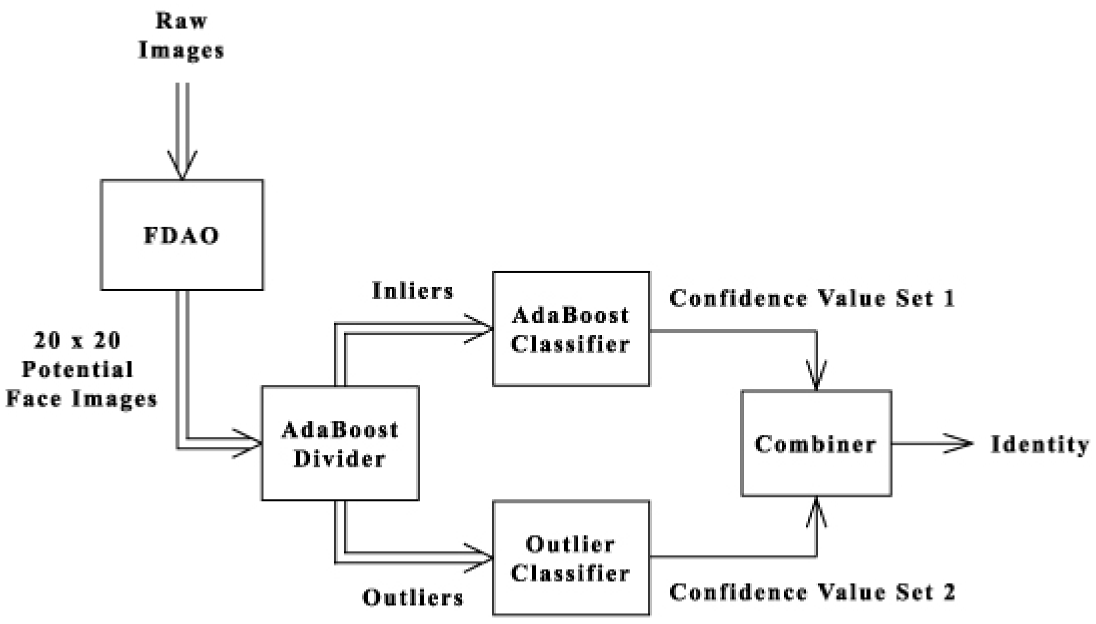

Figure 5.16 The FISA System in the training stage

The Divider is implemented by a back-propagation neural network dividing network (as shown in Figure 5.17) based AdaBoost Machine (with 400 input nodes in the input layer, 15 hidden nodes in the hidden layer and 40 output nodes in the output layer of the base back-propagation neural network classifier) and a gating network similar to the one used in the FDAO system. The output of the Dividing network denotes the confidence values (associated with the corresponding labels) judging the input being the face of a particular candidate (there are 40 candidates in the FISA system). The output confidence values and the expected results are fed to the gating network to do a simple comparison to decide whether the labels match the target ones. If matched, it is an Inlier otherwise it is an outlier. Other implementation parameters in the Divider of the FISA system are similar to those of the Divider of the FDAO system. Cascades of the binary AdaBoost machines are used to implement the multi-value classification systems.

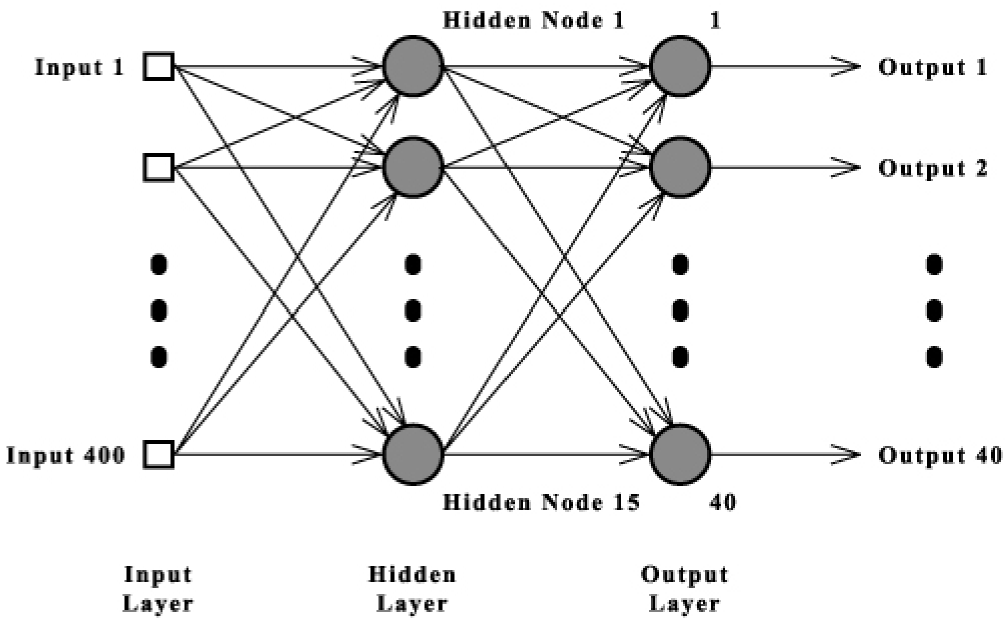

Figure 5.17 The back-propagation neural network dividing network base classifier in the Divider of the FISA system

The output 20 X 20 pixel inliers from the Divider are fed to the Inlier Classifier (whose base learner is implemented by the same kind of back-propagation neural network as that of the FISA Divider), and the Outliers are sent to the Outlier Classifier (implemented by a multi-layer RBF neural network, as shown in Figure 5.18) for classification.

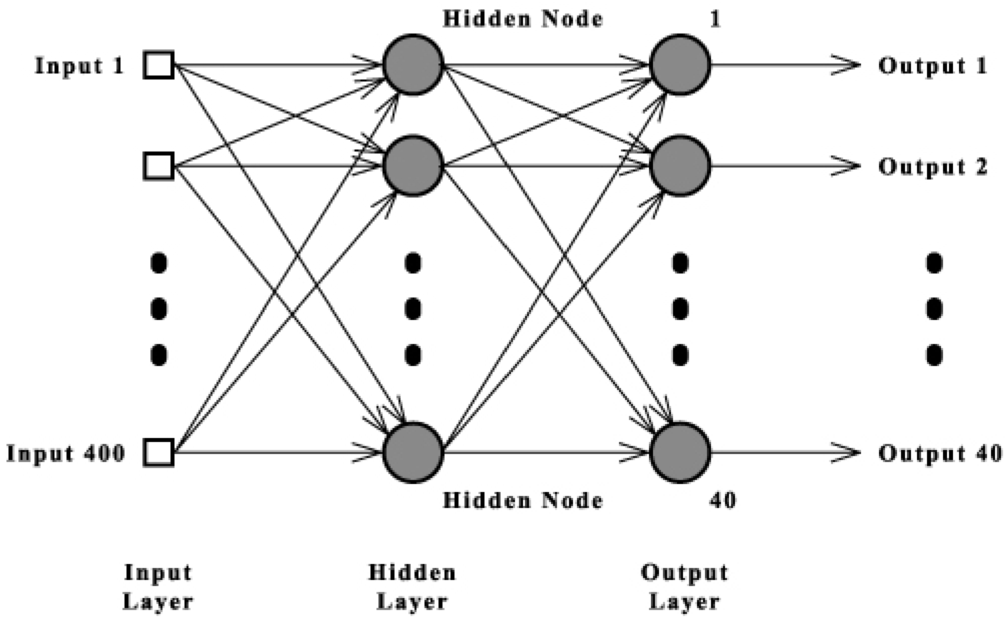

Figure 5.18 The radial basis function neural network outlier classifier in the FISA system

The RBF (Radial Basis Function) neural network is chosen due to its good localization and memory capability. Similar to the back propagation neural network base classifier, there are also 400 input nodes in the input layer in the RBF neural network and 40 output nodes in the output layer. The number of hidden nodes is dynamic and auto-growing depending on the diversity of the training patterns during the training stage. The radii of the hidden nodes in the RBF neural network are also chosen to be very small to enhance RBF network's good local clustering characteristic.

Two sets of confidence values (40 values in one set) (labels are associated with them) act as the outputs of the inlier classifier and the outlier classifier. The confidence values act as the inputs to the combiner (implemented by a three layer back-propagation neural network as shown in Figure 5.19), which generates the final classification result determining the identity of the input image. Multi-layer back propagation neural network with 80 input nodes in the input layer, 15 nodes in the hidden layer and 41 output nodes in the output layer is used to implement the FISA combiner.

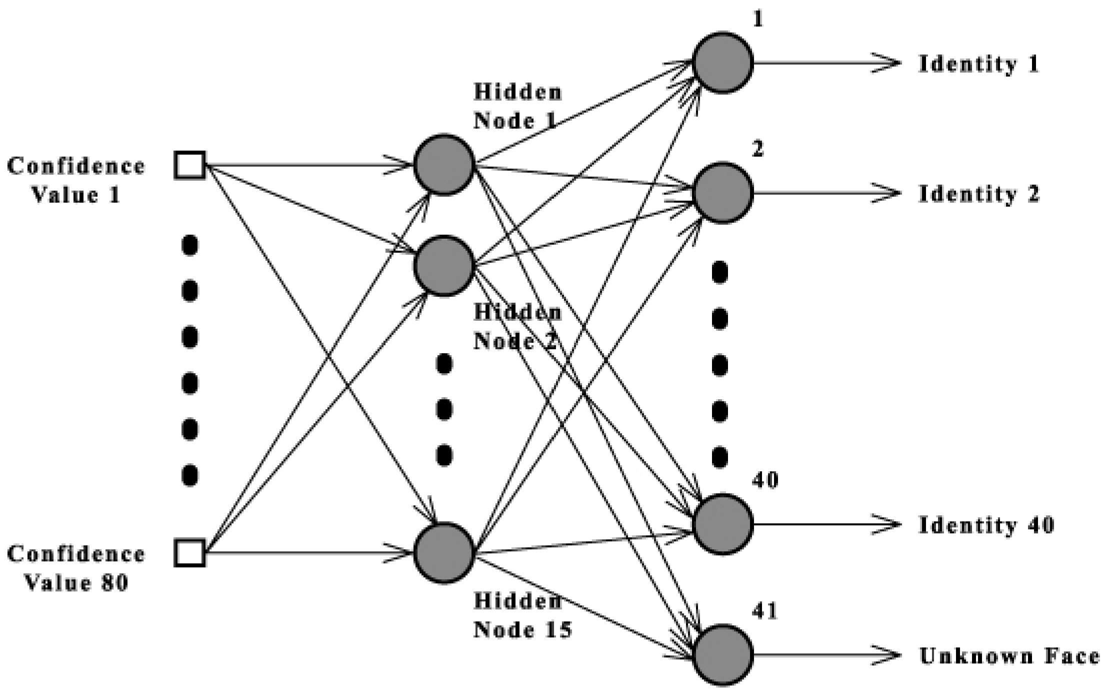

Figure 5.19 The back propagation neural network combiner in the FISA system

5.9.2 The Experimental Results of the FISA System

The same international airport environment selected by the FDAO system is chosen as the complex environment to test the effectiveness of the FISA system. In the FISA system, the face images of 40 candidates are stored in the candidate database. 5000 images with one or multiple face images in an airport environment are collected; about 10% of the images contain one or a few candidate faces. 3000 images are randomly selected as the training set and the remaining 2000 images are chosen as the test set. It is noticed that during the collection period, there are hairstyle and glasses wear changes to the 40 candidates. The FDAO system processes the input images and generates constant flow of 20 X 20 segmented potential face images to the FISA system.

To test the effectiveness of the S-AdaBoost **(S-AB)** algorithm-based FISA system on the face identification application in the airport environment, the performance of the FISA system is compared with those of the other leading approaches. The neural network based **EBGM** (Elastic Bunch Graph Matching) approach [Wiskott, Fellous and Malsburg, 1997], the statistical subspace **LDA** (Linear/Fisher Discriminant Analysis) approach [Zhao, Chellappa and Krishnaswamy, 2000], and the Probabilistic PCA (Principle Component Analysis) approach **PPCA** [Moghaddam, 2002], are implemented. The False Negative Rate (FNR) and the False Positive Rate (FPR) of the four algorithms as well as those of the AdaBoost (**AB**) algorithm are calculated in the experiment. (We thank some facial researchers for providing some of the source codes for the above algorithms.) All approaches are trained and tested on the same dataset collected from the airport. 10-fold cross-validation method is used in all the tests. In the FISA testing system, the 20 X 20 pixel face images detected by the FDAO systems are fed directly to the Inlier Classifier and the Outlier Classifier without passing through the Divider (as shown in Figure 5.20)

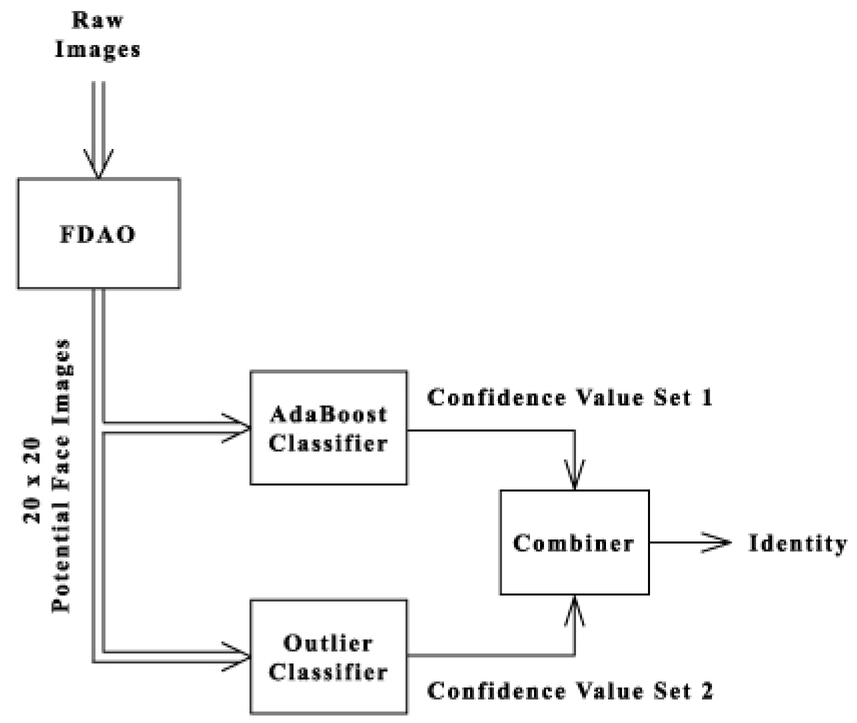

Figure 5.20 The FISA System in the testing stage

The testing results obtained from the various face identification approaches are listed in Table 5.5.

Error Rate %	**S-AB**	**AB**	**EBGM**	**LDA**	**PPCA**
FPR	33.4 ± 4.5	44.6 ± 4.6	40.2 ± 4.3	43.3± 3.8	38.5± 3.2
FNR	0.1± 0	1.4± 0.1	0.2± 0	0.6± 0	0.5± 0.1

Table 5.5: The error rates of different face identification approaches on the airport database

From the experimental results, it is found that the S-AdaBoost algorithm-based FISA system achieves the best result among all the approaches on the complex airport database both in terms of the FPR and the FNR. The other approaches seem to suffer more when the datasets are "noisy" with a large amount of outliers. The "divide and conquer" approach used in the S-AdaBoost algorithm shows its effectiveness in this round of tests in the face detection application in the complex airport environment.

The FERET [NIST, 2001] (FacE REcognition Technology) database is the leading testing database for the face identification applications. It consists of 14126 images from 1199 individuals. Experiments are conducted using datasets from the FERET database to compare the results of the above leading approaches. The experimental data consists of the training "gallery" of 150 candidate FERET faces and 240 "probe" images containing one or more views of every candidate in the gallery. The input values to the systems are normalized with zero mean and standard deviation of one. The probe images are of different expression, illumination, and poses, with/without glasses. Due to the limited size of the testing gallery, 10-fold cross-validation method is used in the experiment. The experimental results on this standard database are listed in Table 5.6.

Error Rate %	**S-AB**	**AB**	**EBGM**	**LDA**	**PPCA**
FPR	5.2 ± 0.4	9.8± 0.6	6.5± 0.4	7.0± 0.4	5.5± 0.4
FNR	0± 0	1.2± 0.1	0.5± 0	0± 0	0± 0

Table 5.6: The error rates of different face identification approaches on the FERET database

From the experimental results in Table 5.6, it is found that the S-AdaBoost algorithm-based FISA system marginally outperforms the other approaches on the datasets. Comparing with the results in Table 5.5, this marginal out-performance in Table 5.5 might due to the relative small Probe images/Candidate images ratio as well as that the FERET dataset used in the experiment is not as “noisy” and “real” as the complex airport environment dataset.

Chapter Six

Conclusion

This chapter summarizes what have been reviewed, proposed, developed, and tested. Conclusions are drawn and future research direction discussed.

6.1 Concluding Remarks

In the thesis, a new ensemble boosting algorithm, S-AdaBoost, is introduced after reviewing the popular adaptive boosting algorithms and the need to improve the outlier handling capability of the current ensemble boosting algorithms in the complex environment. The contribution of the S-AdaBoost algorithm is to make use of the AdaBoost's adaptive distributive weight as a dividing tool to divide the input space into inlier and outlier sub-spaces and to use dedicated classifiers to handle the inliers and outliers in the corresponding sub-spaces before combining the results of the dedicated classifiers.

The S-AdaBoost algorithm's effectiveness is demonstrated by the experimental results conducted on some benchmark databases through comparing with other leading outlier handling approaches. To further demonstrate the effectives of the S-AdaBoost algorithm in the real world environment, two application systems, the face detection system FDAO and face identification system FISA are developed. FDAO system's performance is compared with the leading face detection approaches using the data obtained from both the complex airport environment and some popular face database repositories. The experimental results demonstrate the effectiveness of the S-AdaBoost algorithm on the face detection application in the real airport environment. Similarly, the FISA system is also based on S-AdaBoost algorithm and its performance is compared with the leading face identification approaches using the airport data and the FERET standard dataset. The results obtained are equally promising and convincing, which demonstrate that the S-AdaBoost algorithm is effective in handling the complex real airport environment in the face identification application.

6.2 Future Research

While the S-AdaBoost algorithm and the systems build on the S-AdaBoost algorithm present positive and promising results, it also generates many questions to be answered in the future.

1. The theory foundation of the S-AdaBoost algorithm is to be enhanced. Even though many experimental results demonstrate the effectiveness of the algorithm, more theoretical analysis is to be conducted in the future.
2. To apply the same Divide and Conquer strategies used in S-AdaBoost to other machine learning algorithms to enhance their outlier handling capability in the real world environment. In order to do that, the most crucial issue is to find some parameters, which function as the "distributive weight" in AdaBoost, in those algorithms to divide the Input Space.
3. The setting of the threshold value in S-AdaBoost algorithm needs to be studied further so as to provide a better understanding and guideline to the design of the S-AdaBoost based application systems.
4. Another research direction is to integrate the understanding of structure and features of the patterns to be recognized into the design of S-AdaBoost

algorithm. A variant of the S-AdaBoost algorithm, Sub-Space AdaBoost, is being developed along this direction.

Appendix A

Bias/Variance Analysis of the ensemble algorithms

According to the statistical learning theory (Vapnik, 1995), in the neural network environment, the learning is a process focusing on reducing the deviation between a targeted function f(x) and the real function F(x,w), which is realized by the neural network architecture. The vector x denotes the input signal and the vector w denotes the weights of the neural network. A neural network is a mechanism being able to incorporate the empirical knowledge of the phenomena through learning. For a stochastic phenomenon described by a random vector X (consisting of some independent variables) and a random dependent scalar D (consisting of some independent variables), the realization of the vector X and the dependent scalar D are denoted by $\{x_i\}$ and $\{d_i\}$. The realization of the phenomenon is denoted by $\acute{R}= \{(x_i,d_i)\}$. As we might not be able to obtain the absolute accurate relationship between the vector X and the dependent scalar D from the learning itself, we have:

$$D = f(X) + \varepsilon \qquad \text{(A.1)}$$

Where ε denotes the neural network's ignorance of the dependency between the vector X and the dependent scalar D. The parameter ε is defined as the expectation error with zero mean and a positive probability of occurrence:

$$\tilde{E}[\varepsilon|x] = 0 \qquad \text{(A..2)}$$

Where Ẽ is the expectation operator. From A.1 and A.2, we can conclude that:

$$f(x) = \tilde{E}[D|x] \qquad \text{(A.3)}$$

$$\tilde{E}[\varepsilon f(x)] = \tilde{E}[\tilde{E}[\varepsilon f(x)|x]$$

$$= \tilde{E}[f(x)\tilde{E}[\varepsilon|x]$$
$$= \tilde{E}[f(x)\ 0]$$
$$= 0 \qquad (A.4)$$

The expectation of the squared distance (or the mean squared error) between the targeted function f(x) and the real function F(x,w) in classification is (according to A.3 and A.4):

Mean Squared Error

$$= SUM_x\,[f(x) - F(x, \acute{R}))^2]$$
$$= \tilde{E}_{\acute{R}}\,[(f(x) - F(x,w))^2]$$
$$= \tilde{E}_{\acute{R}}[(\tilde{E}[D|X=x]- F(x, \acute{R}))^2] \qquad (A.5)$$

The above equation can be understood as that the mean squared error is equal to the average mean of the estimation error between the function f(x) and the approximated function F(x, Ŕ). In the following equation, an item $\tilde{E}_{\acute{R}}[F(x, \acute{R})]$, which is the average expectation over all of the training patterns, is inserted:

Mean Squared Error

$$= \tilde{E}_{\acute{R}}\,[(\tilde{E}[D|X=x]- F(x, \acute{R}))^2]$$
$$= \tilde{E}_{\acute{R}}\,[(\tilde{E}[D|X=x] - \tilde{E}_{\acute{R}}[F(x, \acute{R})]) + (\tilde{E}_{\acute{R}}[F(x, \acute{R})] - F(x, \acute{R})))^2]$$
$$= \tilde{E}_{\acute{R}}\,[(\tilde{E}[D|X=x] - \tilde{E}_{\acute{R}}[F(x, \acute{R})]))^{2} + ((\tilde{E}_{\acute{R}}[F(x, \acute{R})] - F(x, \acute{R})))^2 + 2(\tilde{E}[D|X=x] - \tilde{E}_{\acute{R}}[F(x, \acute{R})])\,(\tilde{E}_{\acute{R}}[F(x, \acute{R})] - F(x, \acute{R}))]$$
$$= \tilde{E}_{\acute{R}}\,[(\tilde{E}[D|X=x] - \tilde{E}_{\acute{R}}[F(x, \acute{R})]))^{2}]+ \tilde{E}_{\acute{R}}\,[((\tilde{E}_{\acute{R}}[F(x, \acute{R})] - F(x, \acute{R})))^2] + 2\tilde{E}_{\acute{R}}\,[(\tilde{E}[D|X=x] - \tilde{E}_{\acute{R}}[F(x, \acute{R})])\,(\tilde{E}_{\acute{R}}[F(x, \acute{R})] - F(x, \acute{R}))]$$

According to (A.4),

Mean Squared Error

$$= \tilde{E}_{\acute{R}}\,[(\tilde{E}[D|X=x] - \tilde{E}_{\acute{R}}[F(x, \acute{R})]))^{2}]+ \tilde{E}_{\acute{R}}\,[((\tilde{E}_{\acute{R}}[F(x, \acute{R})] - F(x, \acute{R})))^2]$$
$$= Bias^2(w) + Variance(w) \qquad (A.6)$$

Bias is defined as:

$$\text{Bias} \equiv \tilde{E}_{\acute{R}} [\tilde{E}[D|X=x] - \tilde{E}_{\acute{R}}[F(x, \acute{R})]]$$

Bias denotes the inability of the individual neural network component to accurately measure the decision boundary; it is also called the approximation error. Variance is defined as:

$$\text{Variance} \equiv \tilde{E}_{\acute{R}} [((\tilde{E}_{\acute{R}}[F(x, \acute{R})] - F(x, \acute{R})))^2]$$

Variance denotes the inadequacy and the limitation of the training pattern set, it is also called the estimation error. Differentiating these two kinds of errors will help us to analyze the effectiveness of the ensemble machines. It is noticed that in a single neural network system with a fixed size of training set, if we want to reduce bias, the variant often goes up; if we want to reduce variant, we need to pay the price of bias going up. It is only possible to reduce both bias and variance when we have infinite number of good training patterns (Geman, Bienenstock and Doursat, 1992), this phenomenon is called the bias/variance dilemma. To tackle this dilemma, at least two approaches have been proposed. One way to circumvent this is to introduce "harmless" bias to maintain bias and reduce variance [LeCun 1990], another way is to maintain bias and reduce variance using ensemble approaches [Naftaly, Intrator and Horn,1997]. The S-AdaBoost ensemble method falls into the second category.

In the S-AdaBoost algorithm based classifiers, differently trained classifiers (we use the neural network classifiers in this section for discussion) share different distributions of the training input patterns; the classification results (confidence values) of these classifiers are combined to produce the final classification output. It is noticed that if all those individual classifiers were combined to form one big classifying neural network, the number of free parameters to be decided would be numerous, which could easily lead to the overfitting of the big network. In the ensemble approach, each component classifier is trained separately; the chance of overfitting is thus reduced.

In the ensemble approach, it is expected that each hypothesis classifier focuses on its own domain and converges to its own local minimum of the cost function. It is hoped that in this way the combiner can boost the combined performance of the ensemble.

In an ensemble, the Space **Ž** is defined as the (production of) Input Space **Ŝ** together with the Parameter Space **Č**, which stands for all the training parameters (including the distribution parameter and the network parameter).

$$\check{\mathbf{Z}} = \hat{\mathbf{S}} \text{ X } \check{\mathbf{C}} \qquad \text{(A.7)}$$

Each member of the ensemble has its own Parameter Space **Č**. There are many distribution methods available for selection. In the following discussion, simple averaging is used as the representative combination function for discussion. Defining:

$$F_{av}(x) = \text{Mean of the } F(x) \text{ over the Parameter Space } \check{\mathbf{C}} \qquad \text{(A.8)}$$

So, we have:

$$\tilde{E}_{\check{Z}} [(\tilde{E}[D|X=x]- F(x))^2] = \text{Bias}_{\check{Z}}^2(F(x))+ \text{Variance}_{\check{Z}} (F(x)) \qquad \text{(A.9)}$$

$$\text{Bias}_{\check{Z}}^2(F(x)) = ([(\tilde{E}[D|X=x]]- \tilde{E}_{\check{Z}} [F(x)])^2 \qquad \text{(A.10)}$$

$$\text{Variance}_{\check{Z}} (F(x)) = \tilde{E}_{\check{Z}} [(\tilde{E}_{\check{Z}} [F(x)]- F(x))^2] \qquad \text{(A.11)}$$

Combining (A.8) and (A.9), we have

$$\tilde{E}_{\check{C}} [(\tilde{E}[D|X=x]- F_{av}(x))^2] = \text{Bias}_{\check{C}}^2(F(x))+ \text{Variance}_{\check{C}} (F(x)) \qquad \text{(A.12)}$$

$$\text{Bias}_{\check{C}}^2(F(x)) = ([(\tilde{E}[D|X=x]- F_{av} (x))^2 \qquad \text{(A.13)}$$

$$\text{Variance}_{\check{C}}\ (F(x)) = \tilde{E}_{\check{C}}\ [\tilde{E}_{\check{C}}\ [(F_{av}\ (x)] - F(x))^2] \qquad (A.14)$$

In the Input Space **Ŝ**, we also have:

$$\tilde{E}_{\hat{S}}\ [(\tilde{E}[D|X=x]- F_{av}(x))^2]\ = \text{Bias}_{\hat{S}}^2(F(x))+ \text{Variance}_{\hat{S}}\ (F(x)) \qquad (A.15)$$

$$\text{Bias}_{\hat{S}}^2(F_{av}\ (x)) = ([(\tilde{E}[D|X=x]- F_{av}\ (x))^2 \qquad (A.16)$$

$$\text{Variance}_{\hat{S}}\ (F_{av}\ (x)) = \tilde{E}_{\hat{S}}\ [\tilde{E}_{\hat{S}}\ [(F_{av}\ (x)] - F(x))^2] \qquad (A.17)$$

From (A.7), we can find that:

$$\tilde{E}_{\hat{S}}\ [(F_{av}\ (x)] = \tilde{E}_{\check{Z}}\ [F(x)] \qquad (A.18)$$

So, (A.16) can be re-written as:

$$\begin{aligned}\text{Bias}_{\hat{S}}^2(F(x)) &= (\tilde{E}_{\hat{S}}\ [(\tilde{E}[D|X=x]- F_{av}\ (x))^2\\ &= (\tilde{E}_{\check{Z}}\ [\tilde{E}[D|X=x]- F(x)])^2\\ &= \text{Bias}_{\check{Z}}^2(F(x)) \qquad (A.19)\end{aligned}$$

As the variant of the F_{av} (x) is equal to the Mean Square Error minus the Bias Square. Based on (A.18) and (A.17), we have:

$$\begin{aligned}\text{Variance}_{\hat{S}}\ (F_{av}\ (x)) &= \tilde{E}_{\hat{S}}\ [(F_{av}\ (x)^2] - (\tilde{E}_{\hat{S}}\ [(F_{av}\ (x)])^2\\ &= \tilde{E}_{\hat{S}}\ [(F_{av}\ (x)^2] - (\tilde{E}_{\check{Z}}\ [(F(x)])^2 \qquad (A.20)\end{aligned}$$

Similarly, based on (A.11) and (A.17), we have:

$$\text{Variance}_{\check{Z}}\ (F(x)) = \tilde{E}_{\check{Z}}\ [(F(x)^2] - (\tilde{E}_{\check{Z}}\ [(F(x)])^2 \qquad (A.21)$$

It is obvious that the mean square error of the F(x) over entire space **Ž** is equal to or greater than that of the F_{av} over the input space **Ŝ**, which means:

$$\tilde{E}_{\check{Z}}\ [(F(x)^2] \geq \tilde{E}_{\hat{S}}\ [(F_{av}\ (x)^2] \qquad (A.22)$$

Combining (A.6), (A.19), (A.20), (A.21), and (A.22), we have

$$\text{Mean Square Error of } F_{av}\ (x) \text{ on } \hat{\mathbf{S}} \leq \text{Mean Square Error of } F(x) \text{ on } \check{\mathbf{Z}} \qquad (A.23)$$

For a simple ensemble machine with averaging combination, we conclude that the ensemble can help to reduce the overall error rate. In the S-AdaBoost machine, the classifiers are extended to different types and the combination method is expanded to be non-linear, which can further regulate the bias/variant trade-off and error rate.

References

1. Ali, K. M., and Pazzani, M. J. (1996). Error reduction through learning multiple descriptions. Machine Learning, 24 (3), 173-202.
2. Allwein E.L., Schapire R.E., and Singer Y. (2000). Reducing multiclass to binary: A unifying approach for margin classifiers. *Journal of Machine Learning Research, 1:113-141.*
3. Anthony M., and Biggs N. (1992) Computational Learning Theory. Cambridge: Cambridge University Press.
4. Atick J., Griffin P. and Redlich N. (1996). Statistical Approach to shape from shading: Reconstruction of three-dimensional face surfaces from single two-dimensional images. Neural Computation. Vol. 8, pp. 1321-1340.
5. Belhumeur P.N and Kriegman D.J. (1997). What is the Set of Images of an Object Under All Possible Lighting Conditions? Proceeding of the Conference on Computer Vision and Pattern Recognition, San Juan, PR, pp.52-58.
6. Belhumeur P.N. Hespanha J.P and Kriegman D. J (1997) Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection. IEEE Transaction on PAMI vol. 19, No. 7, pp. 711-720.
7. Bellman R. E. (1961). Adaptive Control Processes: A Guided tour, Princeton: Princeton University Press.
8. Biederman I. and Kalacsai P. (1998) Neural and Psychophysical Analysis of Object and Face Recognition. Face recognition, from Theory to Applications. Berlin: Springer Verlag, pp. 3-25.
9. Bishop C. M. Neural Networks for Pattern Recognition. Oxford University Press, 1995.

10. Blumer, A, Ehrenfeucht A., Haussler D. and Warmuth M.K. (1989) Learnability and the Vapnik-Chervonenkis Dimension. Journal of the Association for Computing Machinery, vol. 36, pp. 929-965.

11. Breiman L. (1997) Prediction games and arcing algorithms, technical Report 504, Statistics Department, University of California, Berkeley.

12. Breiman L. (1999). Prediction games and arcing algorithms. *Neural Computation,* 11(7):1493-1518.

13. Breiman, L. (1996). Bagging predictors. *Machine Learning, 24,* 123-140.

14. Breman L., Friedman J., Olshen J., and Stone C. (1984) Classification and Regression Trees. Wadsworth.

15. Bridle J.S. (1990) Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition, Neuro-computing: Algorithms, Architectures and Applications, F. Fougelman-Soulie and J. Herault, eds. New York: Springer-Verlag.

16. Bruce V., Hancock P.J.B. and Burton A.M. (1998) Human Face Perception and Identification. Face recognition, from Theory to Applications. Berlin: Springer Verlag, pp. 51-72.

17. Carlos Domingo and Osamu Watanabe (2000). MAdaBoost: A Modification of AdaBoost, Proceedings of 13th Annual Conference on Computing Learning Theory, Morgan Kaufmann, San Francisco, pp 180-189.

18. Chakrabarti Soumen, Shourya Roy and Mahesh Soundalgekar (2002). Fast and accurate text classification via multiple linear discriminant projections. VLDB, Hong Kong, August 2002.

19. Chapelle O., Vapnik V., and Weston J. (2000). Transductive inference for estimating values of functions. *Advances in Neural Information Processing Systems,* volume 12, pages 421-428. MIT press.
20. Chen, H., and Liu R.W. (1992). Adaptive distributed orthogonalization processing for principal components analysis, International Conference on Acoustics, Speech, and Signal Processing, vol.2,, pp. 293-296, San Francisco.
21. Cortex, C., and V. Vapnik (1995) Support Vector Network. Machine Learning, Vol. 20, pp.273-297.
22. Craw I., Tock D. and Bennett. (1992) A. Finding Face Features. Proceedings of the Second European Conference on Computer Vision, pp. 92-96.
23. DELVE: Data for Evaluating Learning in Valid Experiments http://www.cs.toronto.edu/~delve/
24. Dietterich, T. G. (2002). Ensemble Learning. *The Handbook of Brain Theory and Neural Networks, Second edition,* (M.A. Arbib, Ed.), Cambridge, MA: The MIT Press.
25. Dietterich, T. G., & Bakiri, G. (1995). Solving multiclass learning problems via error-correcting output codes. *Journal of ArtificialIntelligence Research, 2,* 263-286.
26. Dietterich, T. G., (1997). Machine Learning Research: Four Current Directions *AI Magazine*. 18 (4), 97-136.
27. Domingo C. and Watanabe O. (2000). MadaBoost: A Modification of AdaBoost, Proceedings of 13th Annual Conference on Computing Learning Theory, Morgan Kaufmann, San Francisco, pp 180-189.

28. Drucker H., Schapire, R., & Simard, P. (1993). Boosting performance in neural networks. *International Journal of Pattern Recognition and Artificial Intelligence, 7,* 704-719.

29. Druker H., Cortes C., Jackel L.D., LeCun Y. (1994) Boosting and other ensemble methods Neural Computation, Vol. 6, pp. 1289-1301.

30. Duffy N. and Helmbold D.P. (2000) Leveraging for regression. In *Proc. COLT,* pages 208-219, San Francisco. Morgan Kaufmann

31. Freund Y. & Schapire R. E. (1999). A short introduction to boosting. *Journal of the Japanese Society for Artificial Intelligence, 14,* 771-780.

32. Freund Y. (1995) Boosting a weak learning algorithm by majority, Information Computation, Vol. 121, pp. 256-285.

33. Freund, Y, & Schapire, R. E. (1996a). Experiments with a new boosting algorithm. *Proceedings of the Thirteenth International Conference on Machine Learning* (pp. 148-156).

34. Freund, Y, & Schapire, R. E. (1996b). Game Theory, on-line prediction and boosting. Proceedings of the 9th Annual Conference on Computing Learning Theory, pp. 325-332. ACM Press, New York, NJ.

35. Freund, Y, & Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting, Journal of Computer and System Sciences, vol. 55, pp.119-139.

36. Freund, Y. (1999) An Adaptive Version of the Boost by Majority Algorithm. Proceedings of the Twelfth Annual Conference on Computational Learning Theory

37. Friedman J. (1995). An overview of prediction learning and function approximation. *From statistics to neural networks: theory and pattern recognition applications*, New York: Springer-Verlag.

38. Friedman J. (1999). Greedy function approximation. Technical report, Department of Statistics, Stanford University, February.

39. Friedman J., Hastie T., and Tibshirani R.J. (2000). Additive logistic regression: a statistical view of boosting. *Annals of Statistics,* 2:337-374.

40. Geman, S., Bienenstock, E. and Doursat R. (1992) Neural Network and the Bias/Variance dilemma, Neural Computation Vol. 4, pp. 1-58.

41. Giroshi F., Jones, M., Poggio T. (1995). Regularization theory and neural networks architecture, Neural Computing, 7, 219-269.

42. GMD: Gunnar Rätsch. http://ida.first.gmd.de/~raetsch/data/benchmarks.htm

43. Govindaraju V., Srihari S.N. and Sher D.B. (1990) A Computational Model for Face Location. Proceedings of the Third International Conference on Computer Vision, pp. 718-721.

44. Grove A. J. and Schuurmans D. (1998). Boosting in the limit: maximizing the margin of learned ensembles. 15th National Conference on Artificial Intelligence

45. Hansen, L., & Salamon, P. (1990). Neural network ensembles. *IEEE Trans. Pattern Analysis and Machine Intell., 12,* 993-1001.

46. Hashem S. (1997) Optimal linear combinations of neural networks, Neural Networks, Vol. 10, pp. 599-614.

47. Hastie T., Tibshirani R., and Friedma J. (2001). The Elements of Statistical Learning: data mining, inference and prediction. Springer series in statistics. Springer, New York, N.Y.

48. Haykin S. Neural Networks: A Comprehensive Foundation. Prentice-Hall, second edition, 1998.

49. He. X, Yan S., Hu and Zhang H.J (2003). Learning a locality preserving subspace for visual recognition. International Conference on Computer Vision (ICCV).

50. Herbrich R. and Weston J. (1999) Adaptive margin support vector machines for classification. Proceedings of the Ninth International Conference on Artificial Neural Networks, pages 880-885.

51. Ho T. K., Hull J. J., and Srihari S.N. Decision Combination in Multi Classifier Systems. IEEE Trans. Pattern Analysis and Machine Intelligence, Vol. 16, no. 1, pp. 66-75, 1994.

52. Jacobs D.W., Belhumeur P.N. and Basri R. (1998) Comparing Images under Variable Illumination. Proceeding of the Conference on Computer Vision and Pattern Recognition, pp. 610-617.

53. Jacobs M.A., Jordan M. I., Nowlan S. J. and Hinton G.E.(1991) Adaptive Mixture of Local Experts. Neural Computation, vol 3, pp. 79-87.

54. Jacobs R. A. (1995). Methods for combining experts' probability assessments. *Neural Computation, 7,* 867-888.

55. Jerome Friedman, Trevor Hastie, and Robert Tibshirani (1998). Additive logistic regression: a statistical view of boosting. Stanford University Technical Report.

56. Jiang W. (2001) Some theoretical aspects of boosting in the presence of noisy data. Proceedings of the Eighteenth International Conference on Machine Learning.

57. Jordan M. I., and Jacobs R. A. (1994). Hierarchical mixtures of experts and the EM algorithm. *Neural Computation, 6,* 181-214.

58. Kanade T. (1973). Picture Processing by Computer Complex and Recognition of Human Faces. Ph.D thesis, Kyoto University.

59. Kearns M. and Valiant L.G.(1994) Cryptographic limitations on learning Boolean formulae and finite automata. *Journal of the ACM,* 41(1):67-95, January.

60. Kivinen J. and Warmuth M.K.(1999) Boosting as entropy projection. Proceedings of the 12th Annual Conference on Computational Learning Theory. Pp 134-144.

61. Kjeldsen R. and Kender J. (1996) Finding Skin in Color Images. Proceedings of the Second International Conference on Automatic Face and Gesture Recognition, pp. 312-317.

62. Kolen, J. F., & Pollack, J. B. (1991). Back propagation is sensitive to initial conditions. In Advances in Neural Information Processing Systems, Vol. 3, pp. 860-867 San Francisco, CA. Morgan Kaufmann.

63. Kotropoulos C. and Pitas I.(1997) Rule-Based Face Detection in Frontal Views. Proceedings of International Conference on Acoustics, Speech and Signal Processing. Vol. 4, pp. 2537-2540.

64. Kwok, S. W., & Carter, C. (1990). Multiple decision trees. In Schachter, R. D., Levitt, T. S., Kannal, L. N., & Lemmer, J. F. (Eds.), Uncertainty in Artificial Intelligence 4, pp. 327-335. Elsevier Science, Amsterdam.

65. Lanitis A., Taylor C.J. and Cootes T.F. (1995) An Automatic Face Identification System Using Flexible Appearance Models. Image and Vision Computing, vol. 13, no.5, pp. 393-401.

66. LeCun, Y., Boser B., Denker J.S., Henderson D., Howard R.E., Hubbard W., Jackel L.D. (1990) Handwritten digit recognition with a back-propagation network. Advances in Neural Information Processing, Vol. 2, pp. 396-404, San Mateo, CA: Morgan Kaufmann.

67. Lee, D., & Srihari, S. N. (1995). A theory of classifier combination: The neural network approach. *Proceedings of the Third International Conference on Document Analysis and Recognition (pp.* 42-45).

68. Leung T. K., Burl M.C. and Perona P. (1995) Finding Faces in Cluttered Scenes Using Random Labeled Graph Matching. Proceedings of the Fifth IEEE International Conference on Computer Vision. Pp. 637-644.

69. Lew M. S. (1996) Information Theoretic View-Based and Modular Face Detection. Proceeding of the Second International Conference on Automatic Face and Gesture Recognition, pp. 198-203.

70. Li S.Z., Zhu L., Zhang Z.Q., Blake A., Zhang H.J. and Shum H. (2002) Statistical Learning of Multi-View Face Detection. Proceedings of the 7th European Conference on Computer Vision. Copenhagen, Denmark. May.

71. Liu J. and Loe K.F. (2003a) S-AdaBoost and face detection in complex environment, Proceedings of Computer Vision and Pattern Recognition 2003, pp. 413-418.

72. Liu J. and Loe K.F. (2003b) Boosting Face Identification in Airports, Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence 2003.

73. Liu J., Loe K.F. and Zhang H.J. (2003c) Robust Face Detection in Airports. The special issue for Biometric Signal Processing, EURASIP Journal on Applied Signal Processing.

74. Maclin R. and Opitz D. (1997). An empirical evaluation of bagging and boosting. In *Proceedings of the Fourteenth National Conference on AI,* pages 546-551.

75. Madhvanath, S., & Govindaraju, V. (1995). Serial classifier combination for handwritten word recognition. *Proceedings of the Third International Conference on Document Analysis and Recognition (pp.* 911-914).

76. Mason L. (1999). *Margins and Combined Classifiers.* PhD thesis, Australian National University, September.

77. Mason L., Bartlett P.L. and Baxter J. (1998). Improved generalization through explicit optimization of margins. Technical report, Department of Systems Engineering, Australian National University.

78. Mason L., Baxter J., Bartlett P.L. and Frean M. (2000). Functional gradient techniques for combining hypotheses. In A.J. Smola, P.L. Bartlett, B. Scholkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers,* pages 221-247. MIT Press, Cambridge, MA.

79. McKenna S., Gong S. and Raja Y. (1998) Modeling Facial Color and Identity with Gaussian Mixtures. Pattern Recognition. Vol. 31, no. 12, pp.1883-1892.

80. Moghaddam B. (2002) Principal Manifolds and Probabilistic Subspaces for Visual Recognition. IEEE Trans. On Pattern Analysis and Machine Intelligence, Vol 24, No. 6, June.

81. Naftaly U. N. Intrator and D. Horn (1997) Optimal ensemble averaging of neural networks, Network, Vol. 8, pp. 283-296

82. Neal, R. (1993). Probabilistic inference using Markov chain Monte Carlo methods. Tech. rep. CRG-TR-93-1, Department of Computer Science, University of Toronto, Toronto, CA.

83. Nilsson N. J. (1965). Learning Machines: Foundations of Trainable Pattern Classifying Systems, New York, Macgraw-Hill.

84. NIST (2001) FERET Database. http://www.itl.nist.gov/iad/humanid/feret/, NIST 42(3):287-320, March. Kluwer Academic Publishers

85. Osuna E., Freund R. and Girosi F. (1997) Training Support Vector Machines: An Application to Face Detection. Proceeding of IEEE Conference on Computer Vision and Pattern Recognition, pp. 130-136

86. P´erez-Cruz F., Alarc´on-Diana P.L.,A.Navia-V´azquez,and A.Art´es-Rodr´iguez. (2001) Fast training of support vector classifiers. *Advances in Neural Inf. Proc. Systems,* volume 13, pages 734-740. MIT Press.

87. Parmanto, B., Munro, P. W., & Doyle, H. R. (1996). Improving committee diagnosis with resampling techniques. In Touretzky, D. S., Mozer, M. C., & Hesselmo, M. E. (Eds.), Advances in Neural Information Processing Systems, Vol. 8, pp. 882-888 Cambridge, MA. MIT Press.

88. Pentland A. (2000a). Looking at People, IEEE Transaction on Pattern Analysis and Machine Intelligence, vol.22, no.1, pp. 107-119, Jan.

89. Pentland A. (2000b). Perceptual Intelligence, Communication ACM, vol. 43, no. 3, pp. 35-44.

90. Pentland A. and Choudhury T. (2000). Face Recognition for Smart Environments. IEEE Computer, pp. 50-55.

91. Perrone M.P. (1993). Improving regression estimation: Averaging methods for variance reduction with extensions, to general convex measure optimization, Ph.D thesis, Brown University, Rhode Island.

92. Pigeon S. and Vandendrope L. (1997) The M2VTS Multimodal Face Database. Proceedings of the First International Conference on Audio and Video-based Biometric Person Authentication.

93. Quinlan J. R. (1992) C4.5: *Programs for Machine Learning.* Morgan Kaufmann.

94. Quinlan J. R. (1996) Bagging, boosting, and C4.5. Proceedings of the Thirteenth National Conference on Artificial Intelligence, pp. 725-730.

95. Rajagopalan A. Kumar K., Karlekar J., Manivasakan R., Patil M., Desai U., Poonacha P. and Chaudhuri S. (1998) Finding Faces in Photographs. Proceeding of the Sixth IEEE International Conference on Computer Vision, pp. 640-645.

96. Rätsch G. (1998). Thesis http://www.first.gmd.de/~raetsch/diploma.ps.gz

97. Ratsch G., Demiriz A., and Bennett K.(2002). Sparse regression ensembles in infinite and finite hypothesis spaces. *Machine Learning,* 48(1-3):193-221. Special Issue on New Methods for Model Selection and Model Combination. Also NeuroCOLT2 Technical Report NC-TR-2000-085.

98. Rätsch G., Onoda T., and Müller K.-R. (2001). Soft margins for AdaBoost. Machine Learning Journal, 42(3): 287-320, March. Kluwer Academic Publishers

99. Ratsch G., Smola A.J., and Mika S. (2003). Adapting codes and embeddings for polychotomies. In *NIPS,* volume 15. MIT Press.

100. Rowley H., Baluja S. and Kanade T. (1998) Neural Network-based Face Detection. IEEE Transaction on Pattern Analysis and Machine Intelligence, vol. 20, no. 1, pp. 23-38, Jan.

101. Schapire R. E., and Singer Y. (1998) Improved boosting algorithms using confidence-rated predictions. Proceedings of the 11th Annual Conference on Computational Learning Theory.

102. Schapire R. E., Freund Y., and Bartlett P. (1997). Boosting the margin: A new expanantion for the effectiveness of voting methods, Machine Learning: Proceedings of the 14th International Conference, Nashville, TN.

103. Schapire R.E., Freund Y., Bartlett P., and Lee W. (1998) Boosting the margin: a new explanation for the effectiveness of voting methods. Annals of Statistics 26(5), 1651-1686.

104. Schapire, R. E. (1990). The strength of weak learnability. *Machine Learning, 5,* 197-227.

105. Schapire, R. E. (1997). Using output codes to boost multi-class learning problems. In Proceedings of the Fourteenth International Conference on Machine Learning, pp. 313-321 San Francisco, CA. Morgan Kaufmann.

106. Schapire, R.E. (1992) The Design and Analysis of Efficient Learning Algorithms.PhD thesis, MIT Press.

107. Schneiderman H. and Kanade T. (1998) Probabilistic Modeling of Local Appearance and Spatial Relationships for Object Recognition. Proceeding of IEEE Conference on Computer Vision and Pattern Recognition, pp. 45-51.

108. Schwenk H. and Bengio Y. (1997). AdaBoosting neural networks *Proc. of the International Conference on Artificial Neural Networks,* pages 967-972, Berlin, Springer.

109. Schwenk H. and Bengio Y. (2000). Boosting neural networks. *Neural Computation,* 12(8): 1869-1887.

110. Sebastian H. Seung and Lee D. Daniel (2000) The manifold ways of Perception. 2268-2269, SCIENCE, December 2000.

111. Sergent J. (1986). Microgenesis of Face Perception. Aspects of Face Processing. Dordrecht: Nijhoff.

112. Servedio R.A.(2001). Smooth boosting and learning with malicious noise. In *Proceedings of the Fourteenth Annual Conference on Computational Learning Theory,* pages 473-489.

113. Sim T., Baker S., and Bsat M. (2003). The CMU Pose, Illumination, and Expression Database, IEEE Transactions on Pattern Analysis and Machine Intelligence.

114. Sinha P. (1995) Processing and Recognizing 3D Forms. Ph.D thesis, MIT.

115. STATLOG: The StatLog Repository. http://borba.ncc.up.pt/niaad/statlog/datasets.html

116. Sung K. K. and Poggio (1998) Example-Based Learning for View-Based Human Face Detection. IEEE Transaction on Pattern Analysis and Machine Intelligence. Vol. 20, no. 1, pp39-51, Jan.

117. Tax, D. M. J., Duin, R. P. W., & van Breukelen, M. (1997). Comparison between product and mean classifier combination rules. *Proceedings of the Workshop on Statistical Pattern Recognition.*

118. Turk M. and Pentland A.(1991) Eigenface for Recognition. Journal of Cognitive neuroscience, vol 3, no. 1, pp71-86.

119. Turner, K., & Ghosh, J. (1996). Error correlation and error reduction in ensemble classifiers. *Connection Science,* 8,385-404.

120. UCI: UCI Machine Learning Repository http: //www1.ics.uci.edu/~mlearn/MLRepository.html

121. Valiant L. G. (1984). A theory of the learnable. Communications of the ACM, 27(11): 1134-1142, November.

122. Vapnik, V.N. (1995). The Nature of Statistical Learning Theory, New York: Springer-Verlag.

123. Venkatraman M. and Govindaraju V. (1995) Zero Crossings of a Non-Orthogonal Wavelet Transform for Object Location. Proceedings of IEEE International Conference on Image Processing. Vol. 3, pp.57-60.

124. Vidyasagar M. (1997) A Theory of Learning and Generalization, London: Spronger-Berlag.

125. Viola P., Jones M. (2001). Fast and Robust Classification using Asymmetric AdaBoost and a Detector Cascade. Neural Information Processing Systems.

126. Weston. J. (1999) LOO-Support Vector Machines. Proceedings of the Ninth International Conference on Artificial Neural Networks, pages 727-733.

127. Wiskott L., Fellous J.M., and Malsburg C. van der. (1997) Face Recognition by Elastic Bunch Graph Matching. IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. 19, pp. 775-779.

128. Wolpert D. H. (1992). Stacked generalization, Neural Networks, Vol. 5, pp. 241-259.

129. Wyner A., Kriege A. and Long C. (2001) Boosting Noisy Data. Proceeding of 8th ICML.

130. Yang G. and Huang T. S. (1994) Human Face Detection in Complex Background. Pattern Recognition vol. 27, no. 1, pp.53-63.

131. Yang J. and Waibel A. (1996) A Real-Time Face Tracker. Proceedings of the third workshop on Computer Vision. Pp. 142-147.

132. Yang Ming-Hsuan, Kriegman David, and Ahuja Narendra. (2002). Detecting Faces in Images: A Survey, IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), vol. 24, no. 1, pp. 34-58.

133. Yow K.C. and Cipolla R. (1997) Feature-based Human Face Detection. Image and Vision Computing. Vol. 15, no. 9, pp. 713-735.

134. Zhao W. (1999) Improving the robustness of face recognition. Proceeding of International Conference on Audio and Video-based Person Authentication, pp. 78-83.

135. Zhao W., Chellappa R, and A. Krishnaswamy (2000b). Discriminant Component Analysis for Face Recognition. Proceedings of International Conference on Pattern Recognition

136. Zhao W., Chellappa R, and Krishnaswamy A. (2000) Discriminant Component Analysis for Face Recognition. Proceedings of International Conference on Pattern Recognition.

137. Zhao W., Chellappa R, Rosenfeld A, Phillips P J. (2000) Face Recognition: a literature survey. http://citeseer.nj.nec.com/374297.html.

138. Zhao W., Chellappa R, Rosenfeld A, Phillips P J. (2000a) Face Recognition: a literature survey. http://citeseer.nj.nec.com/3 74297.html.