

Chapter 4

FLD based Gabor-Boosting Facial Feature Selection

This chapter presents the Fish Linear Discriminant (FLD) weak learner based Gabor-Boosting facial feature selection algorithm. The scenario for face recognition in the chapter is face verification. In Section 4.1, face verification is generalised as a two-class classification task. The motivation of proposing the algorithm is given in Section 4.2. In Section 4.3, three schemes for feature pre-selection are introduced. In Section 4.4, FLD weak learner based AdaBoost training is developed for feature selection. Support Vector Machine (SVM) training with respect to selected features, is given in Section 4.5.

4.1 Face Verification

Face verification has its potential applications such as a door entry system associated with a smart card, and airport board line check associated with a passport photograph, etc. In some sense, face verification could be considered as a special scenario of face recognition. However, face verification is different from face identification. In a face identification scenario, given a face image, the system decides whose face is the encounter person by comparing all faces in the database, and identifies whether the face belongs to the person. In

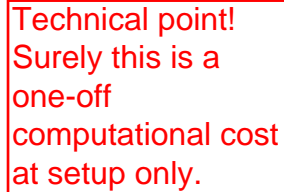
a face verification system, the system is already given the possible identity, and only checks the identity is true or false. For example, in a smart card based authentication system, a chip on the card contains all information of the holder, such as name, date of birth, registration number, etc. Only the smart card holder can possess his or her smart card. When the smart card is read, the system obtains the identity of the holder. Meanwhile a camera takes a snapshot of the holder's face to be compared with face images with the corresponding identity in the database. In general, a face verification system only needs to verify the identity rather than to tell who the identity is. From a technical point of view, face verification avoids searching face database with all candidates. When there is a vast number of users, it could be time consuming.

First use. Spell out

From the point view of classification, face identification can be treated as multi-class classification. Each person can be defined as one unique class in the system. All face images of such a person will be recognised as one class in the system. Most face recognition systems are able to recognise a number (≥ 3) of persons, so that face identification deals with multi-class classification. The classifier in face identification is capable of discriminating multi faces with different criteria. Some approaches such as PCA assume that there is a high-dimensional space from the data vector. The high-dimensional space is called face space, in which it is assumed that there exist many subspaces. Each subspace represents a person. Given a new face image, the identification is to find the subspace where the new face image lay. By finding the subspace of the new image, the subject (or the person) to whom the new image belongs is found.

Face verification can follow the same way as face recognition does. In face verification, there are two classes. One is called client, who is the right person, and the others called impostors. As the identity is already known, the verification is to confirm whether the new image lays in the specified subspace. If yes, it is accepted as a client who has already registered, otherwise it is rejected as an impostor. Consequently, there are only two results, *i.e.*, client or impostor. Therefore, face verification is categorised as a two-class classification. For example, there are four persons registered in a face

Technical point!
Surely this is a
one-off
computational cost
at setup only.



verification system as *client A*, *client B*, *client C* and *client D*. For *client A*, the system accepts face images which represent *client A*, and rejects all other face images (all face images of *client B*, *client C* and *client D*). For each client, a new classifier must be designed. The two-class classification in face verification is more efficient, because the structure of the whole system is simplified to only two subjects. The testing duration is reduced by passing vast searching and matching images across the face image database. Nevertheless, such a face verification system could become computationally heavy when the number of clients is large. It is because a unique classifier has to be built for each client.

4.2 Motivation

Since face verification can be treated as a two-class classification process, some two-class classification approaches can be adapted. Face detection as a well-known two-class classification approach has made rapid progress on the performance. In face detection, there are two classes, *i.e.*, face and non-face. The rapid face detection algorithm was proposed by Viola and Jones [156] in 2001. A machine learning approach has been used for face detection. The approach is capable of processing images extremely rapidly and achieving a high detection rate. The work is distinguished by three key contributions, 1) a new image representation called Haar features; 2) a learning algorithm based on AdaBoost, which selects a small number of critical visual features from a larger set; 3) the detector which is made of cascade structure classifiers, runs at 15 frames per second [156]. In this thesis, some contributions on face detection are adapted in face verification. Face images are represented by some features, like Haar features or Gabor Wavelet Features. AdaBoost selects a small number of features which discriminate clients and impostors in face verification.

4.2.1 Face Verification v.s. Face Detection

Although the AdaBoost algorithm for selecting features is described as object detection in [156], the feature selection approach is very general and can be applied for other objection detection and recognition purposes. Hence, face verification also uses the AdaBoost algorithm to select a small set of significant features representing client, and use the selected features to construct a classifier. In face detection, the final classifier is to distinguish face and non-face, but the final classifier in face verification is to distinguish ~~different individuals' faces~~. The difference between face and non-face is more obvious to human vision perception than face verification. In a machine vision system, when scaling a face image into small size, the face image resembles a horizontal dark bar cross the two eyes and a vertical light column across the nose. By sensing the bar and the column, plausible faces can be detected in images, and non-face images are rejected. In general, decision making in face detection is determined by global variation across ~~over~~ face. However, in face verification, the local variation in faces is very important to contribute to the classification. If face detection and face verification are projected into a face feature space, the distance between a face and ~~an~~ nonface object is larger than the distance between individuals. Therefore, face verification needs more precise description and representation for face images than face detection.

4.2.2 Gabor wavelet feature v.s. Haar feature

Haar features in [156] are used to extract face features, and provide a rich image representation which supports effective learning. Four examples of Haar features are used in [157] as shown in Figure 4.1. The sum of the pixels which lie within the white rectangles are subtracted from the sum of pixels in the grey rectangles. These four Haar features are primitives of all Haar features. All Haar features vary with positions, size of rectangle, and the types. The position determines where the Haar feature is in an image. The size of rectangle determines how large ~~of~~ the Haar feature is. It also can be considered as the scale of Haar feature, because with larger size of rectangles,

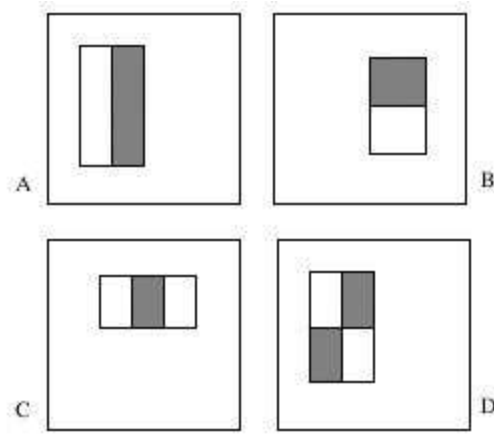


Figure 4.1: The four examples of Haar features. [156]

the Haar feature detects more global variation, whilst with smaller size of rectangles, the Haar features detects more local precise variation. Among these four examples, there are three primitives of Haar features. For (A), the primitive represents features which detect the horizontal variation of pixels in images. For (B), the primitive represents features which detect the vertical variation of pixels in images. (C) can be considered as an alternative of (A), which also detects the horizontal changes. For (D), the primitive is more complicated than others. There are four rectangles in the primitive, and they are in a skew-symmetric structure. The primitive represents features which detect the diagonal variation of pixels in images. Therefore, Haar features contain three kinds of variations: horizontal variation, vertical variation and skew-symmetric variation.

Haar features are similar to Gabor wavelet features. They both have the position variable. The size of rectangle in Haar features is equivalent to spatial frequency ν in Gabor wavelet features, which decides the size of Gabor wavelet kernel. The primitives are equivalent to the orientations in Gabor wavelet features. Due to three primitives, there are three orientations in Haar feature horizontal (0°), vertical (90°) and skew-symmetric (45°). However, Gabor wavelet features can contain any orientation. To detect the salient change of image on different orientations, Gabor wavelet features are

more appropriate than Haar features. In face verification, facial texture is complicated and the salient change could be in different orientations. Gabor wavelet features will provide more precise representation than Haar features do on face verification. In [163], both Gabor wavelet features and Haar features are used on glass detection. The results show that the detection algorithm using Gabor wavelet features have better performance than the detection algorithm using Haar features. Therefore, Gabor wavelet features are used to represent faces in the thesis.

In [140, 168], two similar algorithms of Gabor wavelet features and AdaBoost algorithm are also presented. In these two algorithms, face recognition is not based on classifying subjects, but on classifying *intra-personal difference* and *extra-personal difference* [111] with difference images between two face images. However, human face image appearance has potentially very large intra-personal difference due to facial expression, occlusion (*e.g.* glasses), lighting, pose, and other issues. On the other hand, the extra-personal difference may be small due to the similarity of individual appearance. Figure 4.2 illustrates examples of appearance difference of different subjects. In Figure 4.2, (a) and (b) are images from different persons, but their appearance difference in the input space is smaller than images of the same person but with different poses and lighting changes as shown in (b), (c), and (d). Adini *et al.* [3] demonstrate that the differences between image of a same face due to lighting, viewpoint, and facial expression changes could be larger than those of different faces images. Therefore, the intra-personal and extra-personal recognition mechanism may not be appropriate for face verification.

4.3 Feature pre-selection schemes

As mentioned in Section 3.1.4, a Gabor wavelet feature j is configured by three key parameters, which are the position $z = (x, y)$, the orientation μ , and the spatial frequency ν (also called scale), so that the number of possible Gabor wavelet features is determined by the number of positions, orientations, and scales. It is common to use the parameters as $\nu \in \{-1, \dots, 3\}$,

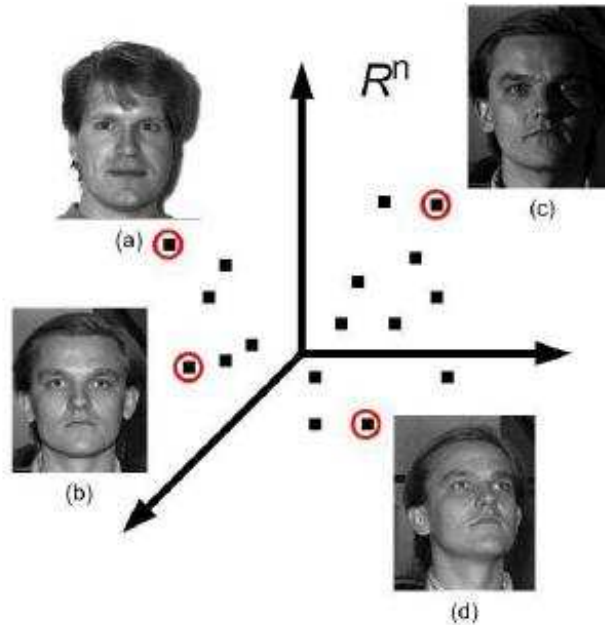


Figure 4.2: The Intra-personal difference versus the extra-personal difference. [101]

and $\mu \in \{0, \dots, 7\}$, and the number of position is due to the size of images processed by the Gabor wavelet transform. If all features are used in feature selection, the process will take very long time to select significant features as face representatives. Three schemes are developed to define the number of positions.

- **Full size** There is no reduction on the number of total features. The number of positions is equal to the original number of pixels;
- **Interlace** The features are sampled column by column and row by row. The number of positions is roughly one quarter of the total number of pixels;
- **Scaling** The response images are scaled down according to the spatial frequency, and the number of positions is reduced.

In **Full size**, the number of positions is same as the number of pixels in images, *e.g.*, if there is an image with 64 pixels in the width and 64 pixels in the height, the number of features is $64 \times 64 \times 5 \times 8 = 163,840$. If the full size scheme is adapted, unless the size of images is comparatively small, the number of features will be resided in a very high level. Large number of features introduces difficulties to the algorithm, *e.g.*, time-consuming and complexity of design. Hence, it is better to reduce the number of features if possible. The other two schemes are grounded on this concern.

The **Interlace** scheme is similar to the Interlace technique [11] for improving the picture quality of a video signal in Television industry. After the Gabor wavelet transform on a single image, there are 40 response images left with the same size as the original image. The scheme is to remove even rows and columns such that only odd rows and columns remain. This makes the number of pixels remained in the response images is roughly one quarter of the original pixels.

The **Scaling** scheme is to reduce the number of features by scaling down these response images with higher spatial frequencies ν . There is a phenomenon that in higher frequency response images, the response images appear blurred. Figure 4.3(a) is a 55×51 XM2VTS face image. Figures 4.3(b), 4.3(c), 4.3(d), 4.3(e) and 4.3(f) are the magnitude of response images at frequency $\nu = \{-1, 0, 1, 2, 3\}$, respectively, and the orientation is $\frac{\pi}{4}$, *i.e.*, $\mu = 2$. As the spatial frequency increases, more blurred are the response images. It indicates that in higher frequency response images, the neighbour features are highly correlated. In Figures 4.3(e) and 4.3(f), human vision perception is unable to distinguish the facial characteristics. In addition, in 2D

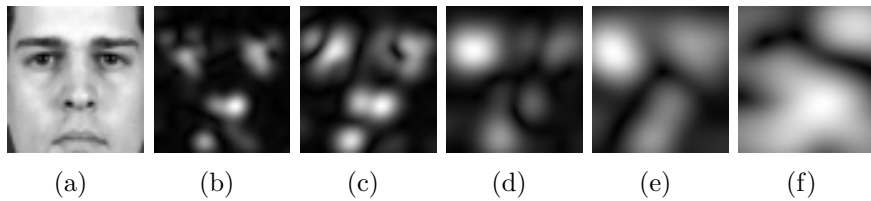


Figure 4.3: The response images from low spatial frequency to higher frequency.

discrete convolution, convolving images with high frequency Gabor wavelet is equivalent to scaling down images and convolving with lower frequency Gabor wavelet as shown in Figure 4.3. In the left of Figure 4.3, the original

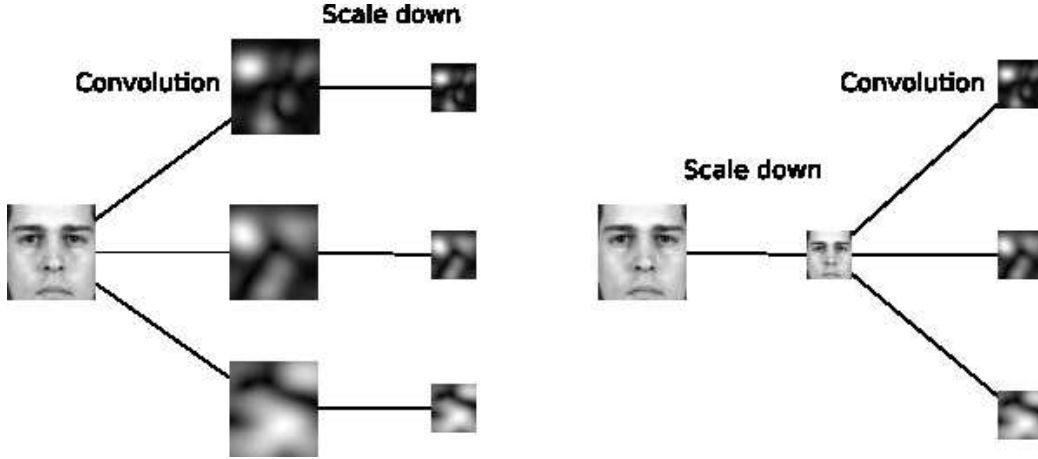


Figure 4.4: The two processes are equivalent in terms of results.

face image is firstly convolved by three Gabor wavelets with $\nu = \{1, 2, 3\}$ to obtain three magnitude responses, then these responses are reduced into the $\frac{1}{4}$ of original size. In the right of Figure 4.3, the original face image is firstly reduced into the $\frac{1}{4}$ of original size, then convolved by three Gabor wavelets with $\nu = \{-1, 0, 1\}$. These magnitude responses are same as those reduced magnitude responses in the left. The results from Figure 4.3 are magnified in Figure 4.5. Figures 4.5(a), 4.5(b) and 4.5(c) are the results from the left part, while Figure 4.5(d), 4.5(e) and 4.5(f) are the results from the right part. The top row and the bottom row in Figure 4.5 show the responses are identical. The processing in which convolution is first done and then scale down, is equivalent to the processing in which scale down is first and then convolution. Figure 4.5 also indicates that higher frequency response images can be scaled down into smaller size without significant information loss

Therefore, in the Scaling scheme, the number of features is reduced by scaling down the response images with higher frequencies. For an image with the size $W \times H$, the response image with the lowest frequency is kept in size

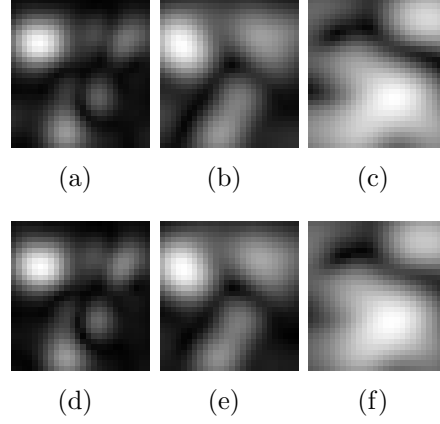


Figure 4.5: Comparison between response images after convolving with high frequency Gabor wavelet and after convolving with low frequency Gabor wavelet.

of $W \times H$. For the second lowest frequency, the size of response images is $\frac{W \times H}{4}$; for the third lowest frequency, the size of response image is $\frac{W \times H}{16}$, and so on. The corresponding number of features are $W \times H \times 8$, $\frac{W \times H}{2^2} \times 8$, $\frac{W \times H}{2^4} \times 8$ and so on. If the width $W = 64$ and the height $H = 64$ with five frequencies $\nu = \{-1, \dots, 3\}$ and eight orientations, the total number of features is

$$(W \times H + \frac{W \times H}{2^2} + \frac{W \times H}{2^4} + \frac{W \times H}{2^6} + \frac{W \times H}{2^8}) \times 8 = 43648$$

With **Full size**, the total number of features is 163,840. In the **Scaling** scheme, the size deduction ratio is $43648/163840 \approx 0.27$.

4.4 Feature Selection

After applying the pre-selection scheme to reduce the feature space, **AdaBoost** algorithm is applied for feature selection. This section presents the algorithm firstly, and then an elementary part of AdaBoost - FLD weak learner is introduced.

Feature selection is a technique, commonly used in machine learning and

pattern recognition for reducing feature space. It selects a subset of relevant features from a large amount of features for constructing robust learning models. By removing irrelevant and redundant features from the training data, feature selection helps in improving the performance of learning models. Feature selection also helps to distinguish features which are important for learning. From a theoretical perspective, feature selection for classification or recognition is to obtain an optimal subset from the whole features against the exhaustive search of all possible subsets.

4.4.1 AdaBoost algorithm for Feature Selection

A variant of AdaBoost is adapted for selecting most important features for face verification. The feature selection algorithm is with respect to individual clients. If the face verification system is required being able to verify all clients, the algorithm will be applied to select most significant features for each client, such that a small number of features corresponding to each client are collected and constructed into a classifier to verify the client. A feature group is unique for a client. The algorithm for feature selection is listed in Table 4.1. In the learning algorithm, the learning process contains n training examples in the training set. Each example (x_i, y_i) is defined as that x_i is the data and y_i is the label. The data of example contains k features $\{j_1, \dots, j_k\}$ which construct a vector with k elements. The label y_i , also called class of the example, 0 or 1 for impostor or client, respectively.

The example is associated with a weight ω to indicate how hard the example for a weak learner is recognised. The allocation of weights on each example is called initialisation. Different from the initialisation in Table 3.1, in this algorithm, client (positive examples) and impostor (negative examples) are given different weights with respect to the number of corresponding examples in the training set. This is due to the fact that the number of client face images may not be equal to the number of impostor face images in the training set. The weight of an example also indicates prior probability which is referred to prior knowledge of the example in the whole population of examples. When positive and negative examples are well balanced, *i.e.*,

Odd word to use.
"well" maybe.

Table 4.1: The variant algorithm of AdaBoost for feature selection

- 1: Given example $(x_1, y_1), \dots, (x_n, y_n)$, where x_i is the data of the i th example, which are contributed by k features $\{j_1, \dots, j_k\}$, and $y_i \in Y = \{0, 1\}$ for impostor and client, respectively.
- 2: Initialise the weights $\omega_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ for $y_i = 0, 1$ respectively, where m and l are the number of impostors and clients, respectively.
- 3: **for** $t = 1, \dots, T$ **do**
- 4: Normalise the weights, $\omega_{t,i} \leftarrow \frac{\omega_{t,i}}{\sum_{i=1}^n \omega_{t,i}}$ so that ω_t form a probability distribution.
- 5: **for all** $\{j_1, \dots, j_k\}$ **do**
- 6: Train a weak learner h_j which is restricted to use a single feature j .
- 7: The error is calculated as $\varepsilon_j = \sum_{i=1}^n \omega_{t,i} |h_j(x_i) - y_i|^2$ with respect to $\omega_{t,i}$
- 8: **end for**
- 9: Choose the optimal weak learner h_t with the lowest error ε_t from all h_j .
- 10: Select the corresponding feature j_t from the weak learner h_t as a significant feature.
- 11: Remove the feature j_t from the feature set $\{j_1, \dots, j_k\}$.
- 12: Update the weights $\omega_{t+1,i} = \omega_{t,i} \beta_t^{1-e_i}$, where $e_i = 0$ if example x_i is classified correctly and $e_i = 1$ otherwise, and $\beta_t = \frac{\varepsilon_t}{1-\varepsilon_t}$.
- 13: **end for**

Doesn't make sense - not sure what it should say.

the numbers are equal, each example will share the same prior probability $\frac{1}{n}$ if there are n examples. When positive and negative examples are not balanced, the prior probability for a particular class is evaluated by the number of examples of the class by the total number of examples. It is assumed that there are l positive examples and m negative examples in the training set, *i.e.*, the whole population $l + m = n$, and both accumulative prior probabilities of positive and negative examples are equal to 0.5. In the perspective of a vector space, the subspace spanned by positive examples takes half capacity of the vector space, and the subspace spanned by negative examples takes another half. Within positive examples, the prior probability of each example should be the same. That is because the examples belonging to the same class are equally of importance and no example in the same class will be

more important than others at the beginning of AdaBoost learning. Hence, if an example is positive, the weight is $\frac{1}{2l}$, otherwise the weight is $\frac{1}{2m}$ at the initialisation.

After initialising the weights, AdaBoost training is going to the stage of iteration running. The number of iterations in the training is determined by the number of expected selected features. If T features are selected from the whole set of features, the number of iterations will be set to T . At the beginning of each iteration, the weights need to be normalised so that all weights form a probability distribution, *i.e.*, the weights are divided by the sum of all weights.

AdaBoost performs an exhaustive search on all features to find the optimal feature. For each feature j , the algorithm trains a weak learner h_j based on the training examples associated with weights. The weak learner h_j is restricted to use one single feature j rather than all features $\{j_1, \dots, j_k\}$, such that the data of each example is represented by only one numeral in the weak learner. This weak learner is called single feature based weak learner. After the weak learner h_j is trained, the error is calculated with respect to the weights. From Table 4.1, it can be seen that the error ε is the sum of the weights whose corresponding example is misclassified. It is obviously that the range of the error is $0 \leq \varepsilon \leq 1.0$.

After the training is done on all single feature based weak learners, the minimum error ε_t among weak learners is obtained by sorting algorithm. The corresponding weak learner h and feature j are retained. The feature j is labelled as j_t , which is the selected feature from the iteration t . The weak learner h_t is the weak learner generated from the same iteration. Also, it is necessary to remove the feature j_t from the whole feature set $\{j_1, \dots, j_k\}$, so that the feature is excluded in the next iteration.

At the end of each iteration, the weights need to be updated according to the performance of the weak learner h_t . The weights are updated in order to emphasise those examples which are misclassified by the weak learner h_t . If an example is classified correctly by h_t , the weight $\omega_{t+1,i}$ will be decreased. If an example is misclassified, the weight will be keep ~~as~~ the same.

4.4.2 Weak Learner: Fisher Linear Discriminant

The choice of weak learner is crucial for the process of AdaBoost training. Some considerations of weak learner are firstly given, then the FLD (Fisher's Linear Discriminant) weak learner is presented.

Weak Learner

Weak learner using single feature is very important for the AdaBoost feature selection. There are many possible choices for weak learners, such as Artificial Neural Network (ANN), Naive Bayes, Fisher's Linear Discriminant, Support Vector Machine (SVM) and so on. The choice of weak learner is based on the two factors:

- The computational cost of weak learner training on a given training set should be low.
- The classification accuracy of weak learner is not necessarily high.

Not clear what this means "be better than random" perhaps?

Firstly, the training time for weak learner should be fast enough, otherwise the overall training of AdaBoost will take extremely long time. For example, training a weak learner requires 1 second on a certain computer, and there are 40,000 features in the training set, so that each iteration will take roughly 11 hours. If the aim is to select 20 features, the total computational time will be about 9 days. Hence, it is important to choose a fast weak learner. Secondly, a learner is defined as "weak" so that classification ability is not expected to be strong. The classification accuracy of a weak learner should normally be better than random guessing, which is 0.5. Hence, an ideal weak learner should be simple and fast. Among those base classifiers mentioned above, a Fisher Linear Discriminant (FLD) weak learner is an appropriate solution in selecting Gabor wavelet features [68, 83, 8]. The FLD weak learner is fast because it is a linear classifier. Non-linearity requires complex structures like SVM and ANN to produce higher accuracy, but penalised by more computational cost. The FLD weak learner does not make the assumption on the distribution of each class, such as Naive Bayes does.

Fisher Linear Discriminant

Fisher's linear discriminant are methods used in statistics and machine learning to find a linear boundary which best separates two or more classes of objects. Fisher linear discriminant was first proposed by R.A. Fisher in 1936 [40]. He defined the separation between two class examples to be the ratio of *the variance between the classes to the variance within the classes*. For a better classification performance, the ratio should be as small as possible. By maximising the variance between classes and minimising the variance ~~between~~ classes, the classifier can reach the optimal performance. When the ratio is at its minimum, the optimal classifier is found.

The input data for weak learners is one-dimensional examples x_1, \dots, x_n , and l is the number of positive examples, and m is the number of negative examples. Each example is labelled with $y = 1$ or $y = 0$ as positive or negative respectively. The purpose is to find a good predictor for the class y of any example. The predictor is defined as

$$y = \mathbf{w}x + b \quad (4.1)$$

\mathbf{w} is the weight coefficient of the predictor, and b is the bias of the predictor. The problem is how to find \mathbf{w} and b . The mean of positive examples and negative examples are

$$\mu_{y=1} = \frac{1}{l} \sum_{y_i=1} x_i \quad (4.2)$$

$$\mu_{y=0} = \frac{1}{m} \sum_{y_i=0} x_i \quad (4.3)$$

so that the variance between classes is $S_b = \mu_{y=1} - \mu_{y=0}$, and the variance within the positive class is

$$S_{w_{y=1}} = \sum_{y_i=1} (x_i - \mu_{y=1})^2 \quad (4.4)$$

and the variance within the negative class is

$$S_{w_{y=0}} = \sum_{y_i=0} (x_i - \mu_{y=0})^2 \quad (4.5)$$

The overall variance within classes is $S_w = S_{w_{y=1}} + S_{w_{y=0}}$. The solution [40] for the coefficient \mathbf{w} optimising the ratio is

$$\mathbf{w} = S_w^{-1}(\mu_{y=1} - \mu_{y=0}) \quad (4.6)$$

The means of projected points are

$$\hat{\mu}_{y=1} = \mathbf{w} \cdot \mu_{y=1} \quad (4.7)$$

$$\hat{\mu}_{y=0} = \mathbf{w} \cdot \mu_{y=0} \quad (4.8)$$

The bias b is calculated as

$$b = -\frac{\hat{\mu}_{y=1} + \hat{\mu}_{y=0}}{2} \quad (4.9)$$

Given a set of examples x_1, \dots, x_n with corresponding label $y = 0, 1$, the training of FLD weak learner is to find \mathbf{w} and b to form a predictor. From the training example, it is easy to computing the means and variances.

4.4.3 Reduction of Computational Time

In AdaBoost, the error of the final strong classifier is bounded by Equation 3.20 in Chapter 3. Thus, there is one hypothesis for AdaBoost.

If weak learners that are slightly better than random guessing can be consistently found, then the error of the final strong classifier drops exponentially fast. [43]

If a weak learner is equivalent to random guessing, its classification error will be 0.5. A weak learner has its error rate more than 0.5, which indicates that given a number of labelled examples, more than half examples are misclassified. It indicates that 0.5 is the limit bounding the error of weak learner.

To have high accuracy in the final classifier, the error of ~~a~~ weak learner should be less than 0.5. Because in the original AdaBoost (in Table 3.1), the weak learner from each iteration is composed into the final classifier. Any weak learner with low classification ability will distort the overall performance of the final combined classifier. Only a weak learner with ~~the~~ performance better than random guessing is allowed to be combined into the final classifier. Although, for each weak learner, the error is expected to be as small as possible, with a simple structure and fast running speed, the weak learners can not be guaranteed to achieve very high classification accuracy.

In AdaBoost based feature selection, low recognition accuracy in a weak learner also indicates that the corresponding Gabor wavelet feature is irrelevant or redundant for face verification. When a weak learner has the error equal or greater than 0.5, AdaBoost training on the weak learner will stop, because the performance of ~~a~~ classifier can not be boosted.

For example, in the experiment which will be described in Section 4.5.2, there are 29,120 features trained in the first iteration. The error distribution of the corresponding 29,120 weak learners are illustrated in Figure 4.6, in which the horizontal bar shows the error and the vertical axis displays the number of weak learners. It can be seen clearly from Figure 4.6, the distribution of the errors from all weak learners looks like a normal distribution with the mean error at 0.35. ~~The most error~~ of ~~a~~ weak learners are in the range ~~from~~ 0.2 to 0.5. There are some weak learners whose errors are equal or greater than 0.5, *i.e.*, error of random guessing. All weak learners having ~~the~~ performance worse than ~~a~~ random guessing will stop in training at the next iteration. The corresponding feature will be discarded at the next iteration to build a weak learner.

The AdaBoost algorithm for feature selection is modified in Table 4.2. In this improved algorithm, a feature set J is defined, which contains all features $\{j_1, \dots, j_k\}$ at the beginning of AdaBoost training. In each iteration, only one significant feature is selected, and some features are removed from the set J . In each iteration, the exhaustive search is on the set J rather than ~~the~~ all features $\{j_1, \dots, j_k\}$.

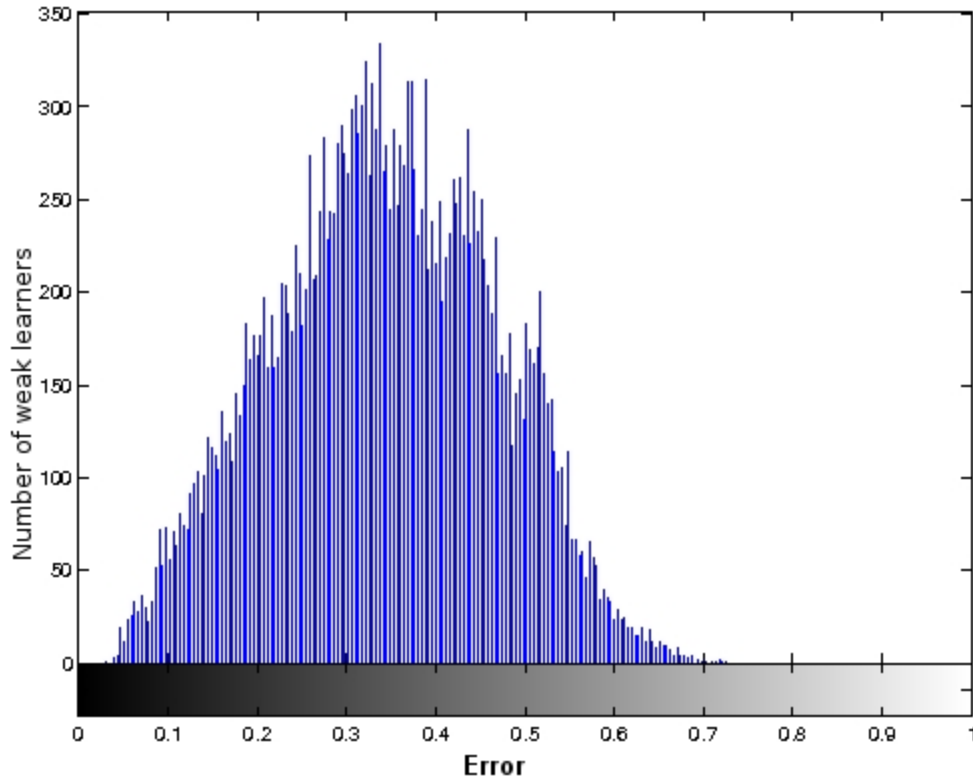


Figure 4.6: The distribution of the error of all weak learners

4.5 Experiments

First use - spell it out here.

The XM2VTS face database is used in the experiments. A small group with 200 face images and a large group with 800 face images of experiments are performed on feature selection. Based on the selected features from the large group of experiments, an SVM classifier is trained for face verification.

4.5.1 XM2VTS Face Database

The developed feature selection algorithm was tested in XM2VTS [108]. The XM2VTS (~~eXtended Multi Modal Verification for Teleservices and Security~~) database is a multi-modal face database which is captured onto high quality digital video. The XM2VTS includes colour images, sound files, video

Table 4.2: The improved algorithm of AdaBoost for feature selection

- 1: Given example $(x_1, y_1), \dots, (x_n, y_n)$, where x_i is the data of the i th example, which are contributed by a set J including k features $\{j_1, \dots, j_k\}$, and $y_i \in Y = \{0, 1\}$ for impostor and client respectively..
 - 2: Initialise the weights $\omega_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ for $y_i = 0, 1$ respectively, where m and l are the number of impostors and clients respectively.
 - 3: **for** $t = 1, \dots, T$ **do**
 - 4: Normalise the weights, $\omega_{t,i} \leftarrow \frac{\omega_{t,i}}{\sum_{i=1}^n \omega_{t,i}}$ so that ω_t form a probability distribution.
 - 5: **for all** j in the set J **do**
 - 6: Train a weak learner h_j which is restricted to use a single feature j .
 - 7: The error is calculated as $\varepsilon_j = \sum_{i=1}^n \omega_{t,i} |h_j(x_i) - y_i|^2$ with respect to $\omega_{t,i}$
 - 8: **if** $\varepsilon_j \geq 0.5$ **then**
 - 9: Remove the feature j from the set J .
 - 10: **end if**
 - 11: **end for**
 - 12: Choose ~~optimal the~~ weak learner h_t with the lowest error ε_t from all h_j .
 - 13: Select the corresponding feature j_t of the weak learner h_t as a significant feature.
 - 14: Select the feature j_t out of the set J .
 - 15: Update the weights $\omega_{t+1,i} = \omega_{t,i} \beta_t^{1-e_i}$, where $e_i = 0$ if example x_i is classified correctly and $e_i = 1$ otherwise, and $\beta_t = \frac{\varepsilon_t}{1-\varepsilon_t}$.
 - 16: **end for**
-

sequences and 3D Models. The XM2VTS contains four sessions of 295 subjects recorded over a period of four months at approximately one month intervals.

In XM2VTS, there are 295 subjects with 200 clients and 95 impostors. Since ~~the~~ frontal face images are ~~only~~ concerned, ~~in~~ each subject 8 up-right and frontal view images have been used. The database was divided into three sets: training set, evaluation set, and testing set. The training set is used to build face models. The evaluation set is for selection of a threshold that determines if a person is accepted or rejected. The testing set is to test the

performance of algorithm. In face verification, a person is accepted if he is a legal client; a person is rejected if he is an impostor. The features are extracted and selected in face images from the training set which includes 200 clients with the first 4 images of each client. In the evaluation set, two face images across 200 clients and 8 images across 20 impostors are collected. In the testing set, two images across 200 clients and 8 images from 75 new impostors are used. The images are selected to bear with moderating differences in illumination, expressions and facial details.

Each subject has 8 frontal images. The images are stored in colour PPM format [160] and at resolution 720×576 . To perform feature selection and face verification, the images are need to be adjusted and segmented, because these large images not only contain inner face area but also the background, the hair, the necks, and so on. The redundant areas such as background, hair, etc, are irrelevant for face verification application. The segmentation to obtain only inner face area is operated. Due to the appearance based approach adopted, for segmenting a face image, some reference points are needed to correspond with all face images. In the thesis, two pupils on face are the reference points to segment inner face images. The following steps are for the processing.

1. If two pupils on a face image are not on the same horizontal line, the rotation of the image will be performed to make two pupils horizontal.
2. The distance between two pupils is measured. The face image is scaled so that the distance between two pupils is 26 pixels.
3. The image is segmented by using a window with 55 pixels in height and 51 pixels in width. The left pupil is matched with one position (13, 19), and the right pupil is matched with another position (39, 19) in the window.
4. The segmented image is converted into a gray-scale image.

Therefore, all images are properly rotated, scaled, segmented and converted to fit a grid size of 55×51 as illustrated in Fig 4.7.



Figure 4.7: The XM2VTS images after rotation, segmentation and scaling

Is this right?

4.5.2 Feature Selection Results

The experiment includes feature extraction and feature selection. The ~~used~~ images are all ~~from~~ the training set. In the training set, there are 4 images per client across 200 clients. 40 Gabor wavelets are generated according to Equation (3.1) with $\nu \in \{0, \dots, 4\}$, and $\mu \in \{0, \dots, 7\}$. Each image is convolved with these 40 wavelets. The outputs are 40 magnitude responses to each image. Each point in a magnitude response image corresponds to a feature. The magnitude response is a map of features' value. The features $O_{\mu,\nu}(z)$ vary with three parameters: orientation μ , scale ν , and position z . For each client, there are $8 \times 5 \times 55 \times 51 = 112,200$ features generated as input for feature selection in each iteration of AdaBoost. The edge of the images is discarded due to edge effect by convolution. The second way is to apply the **Interlace** scheme to reduce the number of features. The actual size of images is reduced to 24×25 . Consequently the number of features for each image is reduced to $8 \times 5 \times 24 \times 25 = 24,000$ instead of 112,200 features.

For each image, there are 24,000 features generated by Gabor wavelet transform. Two groups of experiments are carried out.

- In the first group, a small training set is used. The training set only contains the first 50 subjects of the XM2VTS with 4 images per subject. For each client, 4 images are taken as positive examples, and other 196 images are taken as negative examples.
- In the second group, a large training set is used, which includes the whole 200 clients in the XM2VTS database. For each client, 4 images are taken as positive examples, and other 796 images are taken as negative examples.

This approach gives a large number of training data which are required by AdaBoost algorithm. However it leaves the ratio between positive and negative examples unbalanced, such as 4/196 and 4/796.

Dealing with more than one minimum errors

The feature is selected with the lowest error of the corresponding weak learners. However, in some situation (especially with a small training set) there are more than one weak learners sharing the same lowest error. It gives the problem on how to select features and how to update weights. In the thesis, if the situation occurs, in the current iteration, all features which share the same lowest error are selected as the significant features. The weights are updated according to each selected feature associated with a weak learner separately, and the updated weights are the average weight for all features with the same minimum error. To avoid the problem of multiple lowest errors, the solution is to adapt a larger training set.

odd word. "adopt" maybe?

Result on the small training set

For the small training set, the top 20 features are shown in Figure 4.8. The weak learners associated with these features have the same classification errors, which are equal to zero during AdaBoost feature selection. For the first client, the top 20 features are mostly located on the person's right temple and beside the left eyebrow. Figure 4.8(a) shows these features projecting to the first client face. The locations of the first 20 features (marked with



(a) The 1st client (b) The 4th client

repetition

Figure 4.8: The Gabor wavelet features selected in the first group experiment.

red crosses) selected for the first client by AdaBoost. Most features for the first client lie on a region beside the left eyebrow and a region of right upper temple. These features indicate that the left eyebrow and right-upper temple are significant features for the first client which discriminate the client from others. Most features distributed into two local regions are highly correlated and redundant, as explained in [117]. In the fourth client, the top 20 features locate on the person's cheek near the nostril which appears as a deep valley and nevus on his left face as shown in Figure 4.8(b). These are important signs for the fourth client which discriminate the client from others. These features are also highly correlated.

Result on the large training set

For the large training set, all 800 images from the 200 clients are used in the training set. The AdaBoost algorithm selected the top 20 features from the first client to the 8th client as shown in Figure 4.9. Some features are overlapped because they are sharing the same position, but with different orientation or scale. The first features selected from client 1 to client 8 are shown in the bottom row of Figure 4.9. The results shows that the selected features are randomly distributed in the face area rather than concentrated on some regions of the faces as the results from the small training set. It indicates that these features are less correlated and less redundant than the features selected in the small training set. Most features are located on eyes, eyebrows, nose, lips, glasses frames or beard.

Compared to the high correlation among the features in the small train-

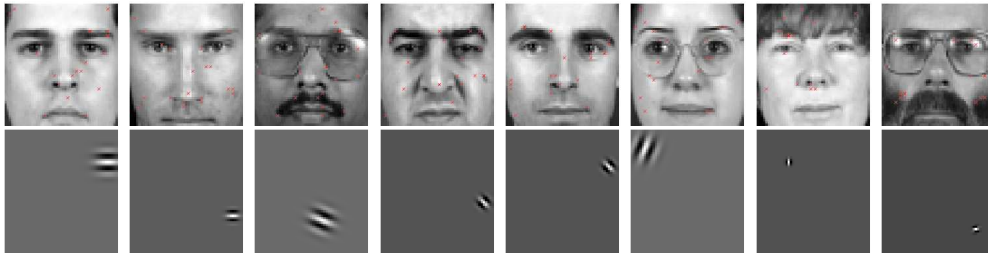


Figure 4.9: The Gabor wavelet features selected after AdaBoost training in the second group experiment

ing set, the features selected from the large training set can be treated as representatives of local regions. The experiments in the large example size separate the features sparsely, and lead to decorrelation and redundancy reduction ~~of output~~. From Figure 4.9, the top 20 features for different individuals are located in different positions. Although most features correspond to the common key characteristics on a face, *e.g.*, eyes, eyebrows, nostril, lips and so on, the top selected features for each individual are varied. As ~~conclu-~~ sion in [117], different subjects can be classified depending on what objects they contain, the faces can be classified depending on the unique characteristics in which an individual is different from others, *e.g.* the nevus on the left face of the fourth client is an unique characteristics for this person. There are some common features selected in both the large and small training sets as shown in Figure 4.10. In Figure 4.10(a), the first selected feature in the large training set are in the eyebrow region, where it is also selected in the small training set. For the fourth client in Figure 4.10(b), the first selected feature corresponds to the nevus on the left face in the small training set. The selected features for the first client and the fourth client in the small training set are shown in Table 4.3 and Table 4.4, respectively. For the first client, most features are distributed into two clusters: one consists of 8 features with the scale and the orientation fitted to 2 and 4 separately.

should this be "in both training sets"?

You lose me here. Why is this relevant? Haven't you already argued that the features selected using the large training set are best - less correlated and less redundant. You don't seem to come to a conclusion here.

Table 4.3: The top 20 features selected by the AdaBoost for the first client in XM2VTS.

Scale ν	Orientation μ	Position (x, y)	Scale ν	Orientation μ	Position (x, y)
1	7	(5,3)	2	4	(49,15)
1	4	(45,13)	2	4	(49,11)
1	4	(47,13)	2	4	(47,15)
1	6	(5,3)	2	4	(45,15)
2	4	(45,11)	0	7	(5,3)
2	4	(45,13)	0	6	(5,3)
2	4	(47,13)	0	6	(3,5)
2	4	(47,11)	1	0	(5,5)
2	4	(49,13)	0	7	(3,3)
1	0	(5,3)	1	6	(39,13)

Table 4.4: The top 20 features selected by the AdaBoost for the fourth client in XM2VTS.

Scale ν	Orientation μ	Position (x, y)	Scale ν	Orientation μ	Position (x, y)
0	0	(34,35)	1	0	(35,36)
0	0	(35,35)	1	0	(36,36)
0	0	(36,35)	1	6	(46,33)
0	7	(36,36)	1	6	(46,34)
1	0	(34,34)	1	6	(47,34)
1	0	(35,34)	1	7	(45,32)
1	0	(34,35)	1	7	(34,35)
1	0	(35,35)	1	7	(35,35)
1	0	(36,35)	1	7	(36,35)
1	0	(34,36)	1	7	(36,36)



Figure 4.10: Common features selected in both the large and small training set

Time reduce

Both algorithms in Table 4.1 and Table 4.2 have been implemented. The algorithm that excludes the big error weak learners displays a significant computational improvement over the original one. The comparison of two algorithms on time cost has been shown in Figure 4.11. In Figure 4.11,

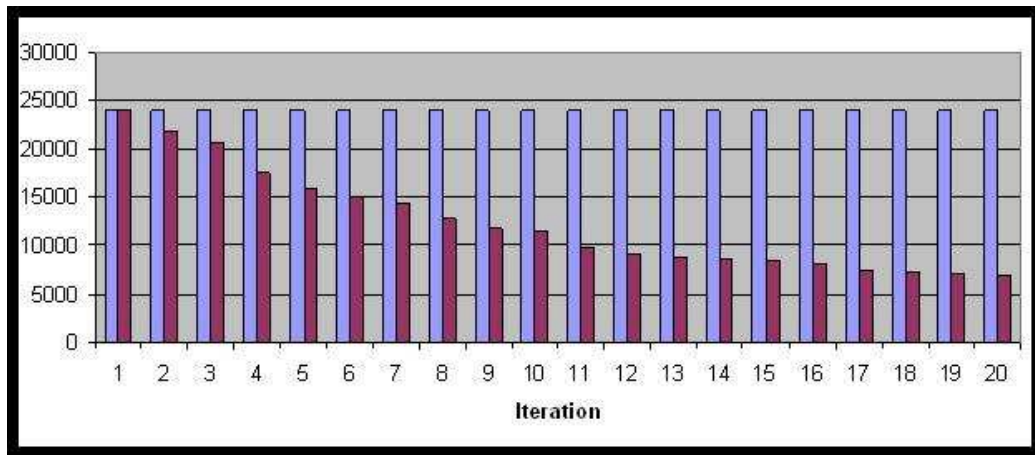


Figure 4.11: The comparison of the improved algorithm and the original algorithm on computational time

the horizontal coordinate indicates which iteration the AdaBoost training is on, and the vertical coordinate shows the number of features trained in the corresponding iteration. The colour lilac shows the number of features without using the computational time reduction technique, and the colour

purple shows the number of features ~~with~~ using the technique. In the first iteration, there are 24,000 Gabor wavelet features ready for AdaBoost training. After the first iteration, 2,177 weak learners ~~has the~~ error greater than 0.5, and the corresponding features are excluded from the set J , such that there are 21,822 features remain for the second iteration. In each iteration, ~~there are~~ some features removed. In the 20th iteration, only 6,194 features ~~are remained instead of 23,981 features in the original algorithm.~~ In total 20 iterations, computational time is reduced to 54.23% of the original time. If there are more than 20 iterations in the training, the computation time will be reduced to much less than 54.23% of the original time. Therefore, the approach which excludes some big error weak learners can reduce the computational cost significantly.

4.5.3 Classification Results

With the 20 Gabor wavelet features selected for each client, a face image is represented by a vector of 20 features. Verification is performed on the training set and the testing set by using a support vector machine (SVM) [114] on each client (subject).

SVM

~~Support vector machines (SVMs)~~ are a collection of learning approaches used for generalised linear classification. A special property of SVMs is that they minimise the classification error and maximise the geometric margin, so that SVMs are also known as maximum margin classifiers. For two-class classification, it is assumed that there is a hyperplane (also called *decision boundary*) separating the clusters of two classified data. SVMs project input vectors to a higher dimensional space where a maximal separating hyperplane is created. The separating hyperplane maximises the distance between the two classes. An assumption is made that the larger the margin or distance between two classes, the lower ~~will~~ the generalisation error of the classifier be. ~~SVM is~~ to maximise the margin between classes and minimise a quantity proportional to the number of mis-classification errors. The original optimal hyperplane

algorithm is a linear classifier. However, non-linear SVMs can be created by applying the kernel method [148] to maximum-margin hyperplanes. The algorithm is allowed to fit the maximum-margin hyperplane in the modified feature space. The classifier is a hyperplane in the high-dimensional feature space, which may be non-linear in the original input space. The kernels can be adopted as a polynomial kernel, a radial based kernel, or a Gaussian radial basis kernel. In this thesis, a Matlab toolbox named OSU Support Vector Machines (SVMs) Toolbox [103] is used to perform the task of classification.

Results

Experiments were carried out on eight clients from the XM2VTS. A non-linear SVM classifier with a polynomial kernel of degree 3 is constructed from the 800 training examples, which are the first four images across all clients. The testing set is composed from the rest four images across all clients and the eight images across all impostors in XM2VTS. The test results of 8 clients with 1,560 testing examples (among them, 4 positive examples and 1,556 negative examples from 199 clients with 4 images and 95 impostors with 8 images per subject) are shown in Table 4.5. By adjusting the bias

Table 4.5: The classification results from client 1 to client 8 (C1 to C8) with 20 features in XM2VTS.

Client	Training Set (%)		Testing Set (%)	
	\mathcal{FP}	\mathcal{FN}	\mathcal{FP}	\mathcal{FN}
1	0.0	0.0	0.0	0.0
2	0.0	0.0	50.0	0.0
3	0.0	0.0	50.0	0.0
4	0.0	0.0	75.0	0.0
5	0.0	0.0	50.0	0.0
6	0.0	0.0	50.0	0.0
7	0.0	0.0	100.0	0.38
8	0.0	0.0	100.0	0.0

Maybe insert a key to show $\mathcal{FP}=\text{False}$ Positive and $\mathcal{FN}=\text{False}$ Negative.

of each SVM classifier for each client, the false positive rate is set to 0%, 25%, 50%, 75% or 100%, and the corresponding classification error (Error

rate) and false negative rates are shown in Table 4.6. The optimal boundary

Table 4.6: The classification results from client 1 to client 8 (C1 to C8) in XM2VTS with shifting bias

False(%) Positive	Error rate(%) / False Negative rate(%)			
	C1	C2	C3	C4
0	4.94/4.95	13.72/13.75	8.53/8.55	2.95/2.96
25	1.73/1.67	12.12/12.08	11.10/8.29	0.26/0.20
50	0.83/0.71	8.72/8.61	9.58/7.13	0.32/0.19
75	0.38/0.19	1.41/1.22	8.30/6.10	0.32/0.13
100	0.26/0.0	0.58/0.32	2.12/1.35	0.26/0.0
	C5	C6	C7	C8
0	2.31/2.31	81.92/82.13	53.08/53.21	15.64/15.68
25	0.45/0.39	14.81/14.78	46.47/46.53	8.59/8.55
50	0.38/0.26	1.86/1.74	44.49/44.47	0.51/0.39
75	0.38/0.19	0.51/0.32	6.47/36.38	0.38/0.19
100	0.32/0.06	0.26/0.0	0.64/0.39	0.32/0.06

for a SVM classifier gives a low error rate but a high false positive rate because the ratio between the positive examples and the negative examples remains unbalanced ¹. Therefore, the decision surface gets close towards the actual boundary of the negative class, but far away from the place where the positive class resides in the feature space. Moreover, SVMs are only concerned with the trade off between margin and mis-classification error, but not with the false positive rate. This leads the trained SVM classifiers having strong ability to recognise the negative examples, but relatively weak ability to recognise the positive examples. It also makes the false negative rate much closer to the classification error rate. By adjusting the bias of the SVM classifier, the false positive rate can be reduced, while the classification error rate is increased, or *vice versa*, e.g. for client 4, the optimal boundary of the SVM classifier gives the false positive rate equal to 25%, and the error rate is 0.26%. However, when the bias of the classifier is adjusted, no false positive is detected, and the error rate is increased to 2.95%. Table 4.6 shows

¹It is also called *selection-bias* or *example-bias*. Section 5.3.2 gives the detail.

that the classification to clients 1, 4 and 5 has better performance than to other clients. It indicates that the 20 selected features contribute well enough for some clients to verification, but they are not sufficient to verify some other clients *e.g.* clients 6 and 7. In this case, it may need more features selected by AdaBoost for client representation.

4.5.4 Discussion on SVMs in Face Verification

The ratio between the number of positive and negative examples is unbalanced. No matter in the small training set or the large training set, the number of positive examples is much less than the number of negative examples. Hence, in the feature space, the subspace which all positive examples dominate is much smaller than the subspace which all negative examples dominate. The negative set contains sufficient data which could describe the generality of all negative examples, however the positive set only has 4 examples, which contains insufficient data to describe itself.

Each example is hypothesised as a point in a two-dimensional feature space as in Figure 4.12. The blue stars and blue circles represent the positive training examples and testing examples, respectively. The red star and red circles represent the negative training and testing examples. The blue solid line denotes the actual boundary for the positive examples, while the blue dash line denotes the predicted boundary for the positive examples. The predicted boundary means that the boundary is predicted by the SVM classifier, not the actual boundary of the examples. As the negative ones, the red solid line and dash line indicate the actual boundary and predicted boundary for the negative respectively. Most negative training examples and testing examples concentrate in the left part of Figure 4.12, while positive examples mass in the right part. The actual boundary for the negative is close to the predicted boundary, which indicates the SVM classifier has achieved a good performance on recognising the negative examples. However, the predicted boundary for the positive is far from the actual boundary, which indicates the bad performance on positive examples. The reason is, the positive training examples are not highly overlapped with the positive testing examples. Some

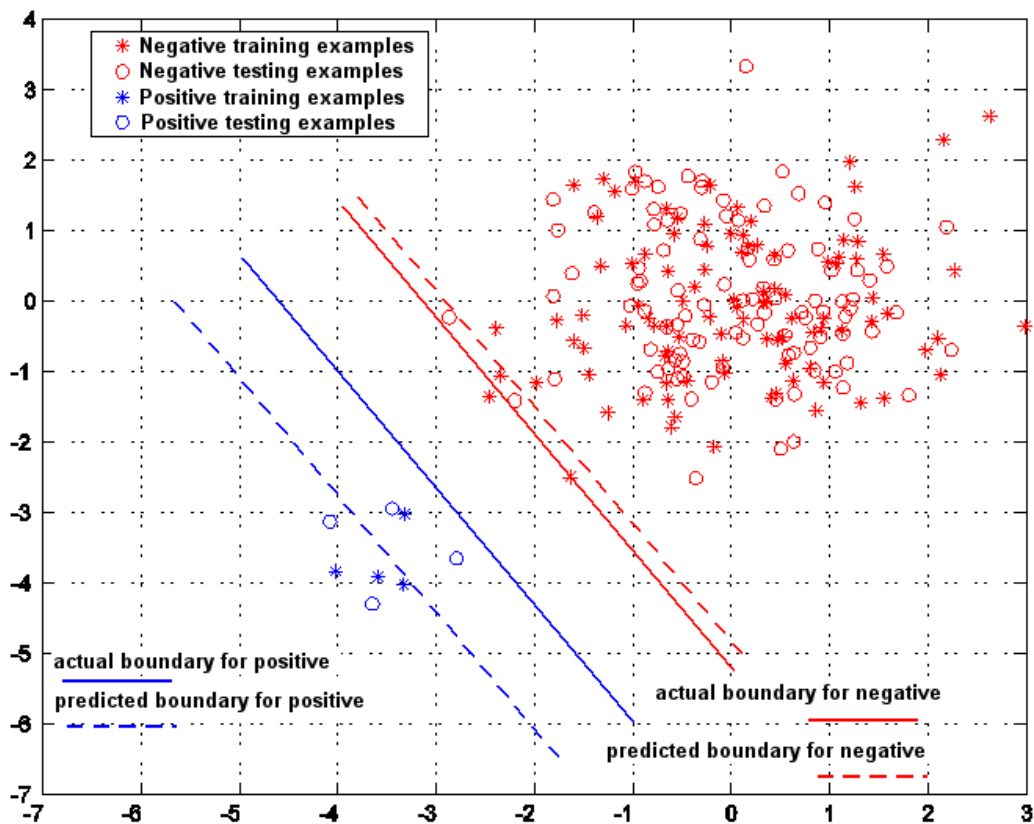


Figure 4.12: The hypothesis on the SVM classification results

positive examples are recognised as non-positive, because they are out of the predicted boundary. To avoid unbalance on positive and negative, increasing new positive examples could be useful.

The size of training data From the perspective of machine learning, to train “ability” on a machine, a data set must be given to the machine to extract information and to learn the relevant decision rules. To build a data set for training (or learning), there is a trade-off between **representatives** of the training set, **conciseness** of the training set and **prior probability** of classes. The representatives of data mean whether the data includes all the possible variants of the described object. The more examples in the data set, the higher the representatives are, and the more the possible variants

This looks like the conclusion but it is not very clear - rewriting this sentence would help a lot.

101

I think you are trying to say that the supplied representative data dictates whether you can cover all the possible variants. The more data - the better

are included. The conciseness of the data refers to the size of the data set. If the size is big, the conciseness of the data set is poor. As long as the examples are sufficiently representative, the number of training examples should be as small as possible. The conciseness of the data is related to computational cost. More data normally leads to more computation time. In supervised learning, the number of examples in a certain class should be in accordance with the prior probability of the class. For example, to recognise a face belonging to male or female, a training set is built with the number of male face images equal to the number of female face images, because it is assumed that the prior probability of male and female are 0.5. In face detection, there are two classes to detect, face and non-face. However, the prior probability of face is far less than the prior probability of non-face, so that the number of face images should be far less than the number of non-face images. Due to the representatives of the data set, the number of face images should not be few, but the set has enough amount to describe all variants of face class. Therefore, to build an appropriate training data set, these three factors, **representatives**, **conciseness** and **prior probability**, should be taken into account, and these three factors should be well balanced.

4.6 Summary

This chapter presents a FLD based Gabor-Boosting feature selection for face verification. The face images in face verification are defined as either clients or impostors, and face verification is treated as a two-class classification task for each client (subject). After Gabor wavelet transform, faces are resided in a very high-dimensional feature space. Feature pre-selection is performed to reduce the feature space to a relatively low-dimensional feature space, and then AdaBoost training is further applied for the purpose. In the AdaBoost training, FLD is used as weak learners, and the strategy is applied for reducing training time. The experiments show the selected features for some clients, and classification performance is tested in the XM2VTS face database. The SVM classifiers are briefly introduced with consideration of balance of examples size. Although the FLD based weak learner displays

appropriate contribution to face verification, the weights for each example are not applied in the training of each FLD weak learner. The next chapter is dedicated to solving the problem.