Chapter 5

Potsu based Binary Gabor Boosting Face Recognition

This Chapter presents a new weak learner named Potsu weak learner in the AdaBoost training for feature selection. Both the XM2VTS and the FERET face databases are tested with the Potsu weak learner based Gabor-Boosting algorithm. Section 5.1 introduces the Potsu weak learner. In Section 5.2, the feature selection of Gabor-Boosting algorithm with the Potsu weak learners is proposed, and optimisation approaches are applied to reduce the computational cost. In Section 5.3, the performance made by the AdaBoost classifier is presented. In the results of classification, overfitting phenomenon is observed. Section 5.4 explores the reasons that cause overfitting. In Section 5.5, two methods are applied on classification to address the overfitting problem. In Section 5.6, the Potsu weak learner is compared with the FLD weak learner with respect to the performance of classification. In Section 5.7, the FERET face database is tested with AdaBoost feature selection of Gabor-Boosting algorithm and SVM classification.

5.1 POTSU Weak Learner

A novel weak learner for AdaBoost training is presented in this section. Firstly, the construction of an ensemble classifier in AdaBoost is explained.

Secondly, the requirement of the training on weak learners in AdaBoost is discussed. Thirdly, the novel weak learner - Potsu weak learner is proposed. The Potsu weak learner benefits from the perceptron prototype which is rapid on training and Otsu's thresholding algorithm which is accurate when the number of examples is large.

5.1.1 Construction of an ensemble classifier

As stated in Section 3.2 in AdaBoost training, corresponding weak learners h_t are collected and built into an ensemble classifier in association with a group of selected significant features. In the original AdaBoost, the ensemble classifier is

$$H(x) = \sum_{t=1}^{T} \alpha_t h_t(x)$$

where, α_t is the importance factor of the weak learner h_t, which indicates how importantly the weak learner h_t contributes to an ensemble classifier, and T is the number of iterations. The importance of a weak learner is evaluated by

$$\alpha_t = \log \frac{1}{\beta_t} \tag{5.1}$$

The parameter β_t is defined as $\beta_t = \frac{\varepsilon_t}{1-\varepsilon_t}$, where the parameter ε_t is the error of each iteration. The importance is also expressed as

$$\alpha_t = \log \frac{1-\varepsilon_t}{\varepsilon_t}$$

Because the error ε_t of weak learner from each iteration should be less than $1/2$, the importance α_t will be greater than zero. The smaller the error ε_t is, the larger is the importance α_t. It indicates that weak learners which have slightly "stronger" discriminating power, play important roles in the ensemble classifier to make decision.

5.1.2 Requirements of weak learners

As discussed in Section 3.2.4, there are two ways to process weak learners. In the first way, the Naive Bayes weak learner is adopted for training and the training data is re-sampled. The size of the new training set is increased ~~to 25 ∼ 42 times of~~ the original size after re-sampling. This requires more computational time. Since AdaBoost training has already inevitably ~~had~~ in higher computational cost, increasing the ~~example~~ size will increase the difficulty on AdaBoost training. Hence the re-sampling approach is not appropriate in the practice of AdaBoost. A better way to implement weak learners is to use the weights directly on the examples, which is the Artificial Neural Network (ANN) applied in Section 3.2.4.

The selection of weak learners is determined by finding the minimum error ε. On each weak learner, the error ε is evaluated with respect to the weights ω of examples, which is

$$\varepsilon = \sum_{i=1}^{n} \omega_{t,i} |h(x_i) - y_i|^2 \qquad (5$$

First 2 sentences don't make much sense - need to rewrite.

Given a set of training examples, training a weak learner is to find a criterion to predict the label of examples. Different criteria could induce different results on labelling the examples. The aim of training a weak learner is to find an optimal criterion with desirable requirements.

Different weak learners, *i.e.*, classifiers, are needed to follow different requirements. Linear Discriminant Analysis (LDA) classifiers require maximising between-class variance and minimising within-class variance. Naive Bayes classifiers use the method of maximum likelihood or maximise a posterior probability. Support Vector Machine (SVM) classifiers are required to maximise the margin between classes and minimise a quantity proportional to the number of mis-classification errors. k-nearest neighbour (k-NN) classifiers are based on the nearest training examples in the feature space. ANN classifiers are required to minimise the ratio between misclassified examples and total examples. For instance, if an ANN classifier is chosen as a weak learner, the error is evaluated by $\frac{1}{n}\sum_{i=1}^{n} |h(x_i) - y_i|^2$ instead of Equation 5.2

without taking weights into account. Therefore the requirement of the ANNs is different from the requirement of weak learners in AdaBoost. In this thesis, weak learners are required to minimise the error ε which is a weighted ratio between misclassified examples and total examples.

It is assumed that training examples for a weak learner ~~are residing~~ in a 2-D feature space[1], where there are many lines across data clusters as shown in Figure 5.1. There are two classes: positive examples represented

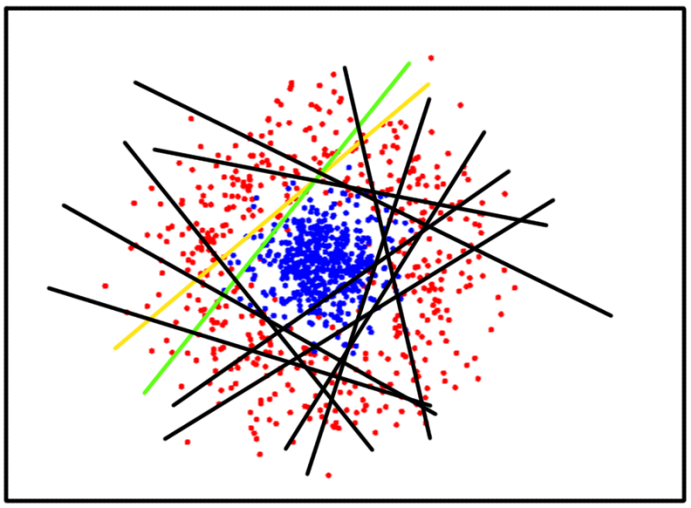


Figure 5.1: Example: a 2-D feature space explains the optimal criterion of weak learner in AdaBoost.

by blue dots and negative examples represented by red dots. Many lines are across the positive data and the negative data, and each line represents ~~each criterion~~ to separate the positive and the negative examples. Among these criteria, there must be an optimal one which minimises the error ε. If an ANN classifier is adopted as a weak learner, the optimal criterion is chosen as the yellow line in Figure 5.1. That is because the optimal criterion in the ANN classifier minimises the ratio between the number of misclassified examples and the number of total examples. Although the yellow line is the best one among other lines for separating the examples, the criterion does not

[1]Actually, the training examples are in a 1D feature space used for face recognition in this thesis.

achieve the minimal ε. The green line in Figure 5.1 is the optimal criterion actually for the weak learner to minimise ε, although there are more examples misclassified by the green line. When those examples are misclassified, the accumulated error ε does not increase significantly since their weights are low. Some examples are misclassified by the yellow line and correctly classified by the green line, since they have higher weights. The accumulated error ε will increase significantly.

The requirement for weak learner in AdaBoost training is that the training criterion must be satisfied with the minimal ε. The classifiers mentioned above, such as LDA, Naive Bayes, SVM, k-NN, and ANN, ~~are not satisfied with~~ the requirement. None of these classifiers can be directly introduced as the weak learner in the training, so ~~that~~ a new weak learner needs to be developed.

5.1.3 Potsu weak learners

A new weak learner for AdaBoost training is proposed in this section. The new weak learner combines the concept of perceptron with Otsu's algorithm, so that the proposed weak learner is named ~~as~~ **Potsu** weak learner. The Potsu weak learners use an example based heuristic approach to threshold the training data.

Perceptron

To train a weak learner rapidly, it is necessary to keep the design of weak learner as simple as possible. Among various prototypes of weak learner, a design of *perceptron* [52] is chosen, as the perceptron is the simplest type of linear classifier.

The perceptron is a type of two-class classifier that maps its input to an output value. A perceptron based weak learner consists of observation f_j on a feature j, a threshold θ_j, and a parity p_j as below

$$h_j(x) = \begin{cases} 1 & \text{if} \quad p_j f_j(x) < p_j \theta_j \\ 0 & \text{otherwise} \end{cases} \tag{5.3}$$

This whole paragraph is not very clear - I can't follow the argument.

~~the~~ observation f_j on a feature j are numerical values, and the perceptron is dealing with data in a~~n~~ one-dimensional space. The parity p_j indicates the direction of the inequality sign, *e.g.*, positive examples may reside above the threshold θ_j in some cases, or in other cases reside below the threshold θ_j. In practice, the parity p_j is either 1 or -1. Finding an appropriate parity is not a difficult task, because there are only two possibilities for parity: either positive examples below the threshold or above the threshold. Given a parity, the difficulty in learning a perceptron is to find the optimal threshold θ_j. A novel algorithm is proposed to find the optimal threshold similar to *Otsu's Thresholding* method.

Otsu's algorithm

The Otsu thresholding method [122] has widely been used in image thresholding in a wide range of applications, such as noise reduction for human behaviour recognition, medical image processing, real-time segmentation of images, document segmentation and so on. Sahoo *et al.* [139]'s study concluded that the Otsu thresholding method was one of the best threshold selection methods for general real world images with respect to uniformity and shape measures. The Otsu thresholding method is based on a very simple idea: finding the threshold that maximises the between-class variance. The Otsu method is optimal for thresholding large objects from the background. It indicates that the Otsu method is fine for thresholding a histogram with bimodal or multi-media distribution. In gray scale image thresholding, the task is to separate the foreground and background. The foreground may be objects ~~need~~ to be tracked or processed, and the background is anything other than the specified objects. For example, in Optical Character Recognition (OCR), binary operation is required for document segmentation. The thresholding method is to separate some pixels belonging to characters and other pixels belonging to ~~white-papers.~~ In gray scale images, pixel values range from 0 to 255 in 256 integers. Each integer is taken as a possible threshold to segment the foreground and the background. For each possible threshold, the between-class variance is computed. From numerous possible

thresholds, the optimal threshold is the one which has the maximum variance. The detailed algorithm for the Otsu thresholding method can be found in [97].

Thresholding algorithm

In the Potsu weak learner, the method is to select a threshold which minimises the error ε_j of a weak learner on the given training examples. Similar to the Otsu method, some possible thresholds are collected beforehand. These possible thresholds are called *threshold candidates*. If there are n non repetitive features, the number of threshold candidates will be $n - 1$. The error ε_j is calculated according to each threshold candidate, finally an optimal threshold is found with the minimum error among the candidates. The algorithm for training a Potsu weak learner is given in Table 5.1.

Parity The parity of Potsu weak learner is determined by comparing the means of all positive examples between all negative examples. If the mean of positive examples is less than the mean of all negative examples, feature values of all positive examples will be below the threshold. It is assumed that the positive examples and the negative examples are separated and normally distributed. The parity is approximately predicted in the training of Potsu weak learners. Since weak learners only have "weak" discriminating power, the parity is not required to be highly accurate. In brief, it is not necessary to implant a "strong" discriminating power in weak learners.

Threshold The training of Potsu weak learner is to label examples. Once the parity is determined, *e.g.*, $p = 1$, the value $f_j(x)$ of the feature j below the threshold θ are classified as ~~the~~ positive, otherwise classified as ~~the~~ negative. The threshold θ should be between the minimum and the maximum of $f_j(x)$. It is in the range of

$$\text{Min}\{f_j(x)\} < \theta < \text{Max}\{f_j(x)\} \tag{5.4}$$

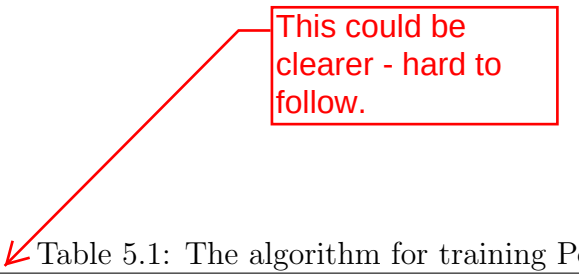
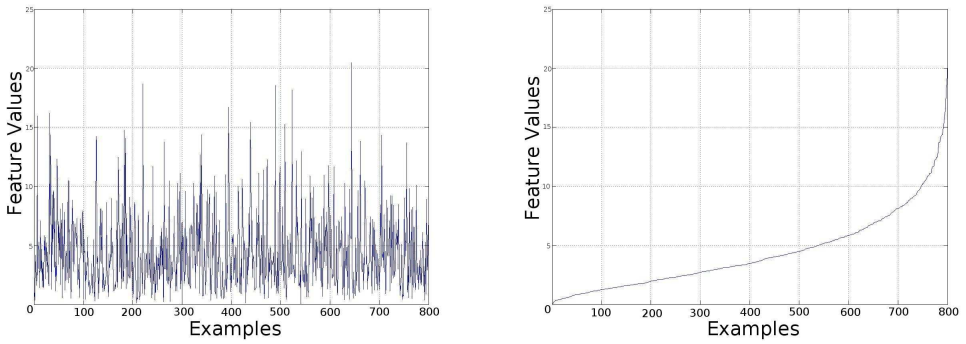


Table 5.1: The algorithm for training Potsu weak learners

1: Given a feature j and example $(x_1, y_1), \ldots, (x_n, y_n)$ and $y_i \in Y = \{0, 1\}$ for negative and positive respectively, $f_j(x)$ is the value of the feature j on the example data x.
2: Construct a Potsu weak learner according to Equation 5.3.
3: Compute the mean of $f_j(x)$ on all positive examples $\bar{\mu}_1$ and the mean of $f_j(x)$ on all negative examples $\bar{\mu}_0$
4: **if** $\bar{\mu}_1 < \bar{\mu}_0$ **then**
5: the parity $p = 1$
6: **else**
7: the parity $p = -1$
8: **end if**
9: Sort all $f_j(x_i) \quad i \in \{1, \ldots, n\}$ into $f'_j(x_i), \ldots, f'_j(x_n)$
10: **for** Every two adjacent sorted examples $f_j(x_i)$ and $f_j(x_{i+1})$ **do**
11: Compute the ith threshold candidates $\theta_i = \frac{f'_j(x_i)+f'_j(x_{i+1})}{2}$.
12: **end for**
13: There are $n - 1$ threshold candidates $\theta_1, \ldots, \theta_{n-1}$.
14: **for all** Threshold candidates $\theta_1, \ldots, \theta_{n-1}$ **do**
15: Suppose θ_i as the threshold θ_j in Equation 5.3.
16: Calculate the error $\varepsilon = \sum_{i=1}^{n} \omega_i |h_j(x_i) - y_i|^2$ with respect to ω_i.
17: **if** $\varepsilon > 0.5$ **then**
18: reject the threshold candidate θ_i.
19: **end if**
20: **end for**
21: Select the corresponding threshold candidate θ with the lowest error ε over all candidates θ.
22: A trained Potsu weak learner is $h_j(x) = \begin{cases} 1 & \text{if} \quad pf_j(x) < p_j\theta \\ 0 & \text{otherwise} \end{cases}$

The threshold θ should be a quantitative value within this range. For instance, there are some examples below a threshold θ' and some examples above. If the threshold θ' has been shifted higher, more examples will be recognised as positive, and vice versa. If the threshold θ' is close to the optimal threshold θ, the error ε' with respect to θ' is also close to the optimal error ε with respect to θ. Since weak learners do not require the "strong" discriminating ~~power. It~~ is necessary to collect a set of possible thresholds, *i.e.*, *threshold candidates* under the framework of Potsu to find a threshold. The performance of weak learners is evaluated on each candidate. Finally, the candidate with the best performance will be selected as the optimal threshold. The number of candidates is required to be large, otherwise the candidates could not cover the whole range of features value. On the other hand, the number should not be too large to compute quickly. The feature values on 800 examples in XM2VTS database are shown in Figure 5.2(a). They are sorted, to be put in order as shown in Figure 5.2(b). It is hypothe-



(a) (b)

Figure 5.2: Feature values on 800 examples: original data and sorted data.

sis that each candidate θ_j exists between two adjacent values. The candidate θ_j is approximated to the middle of these two adjacent values, *i.e.* the mean of the two values. There are $n-1$ candidates $\theta_1, \ldots, \theta_{n-1}$, when there are n training examples existing in the training set. The threshold candidates are displayed in Figure 5.3, in which, the blue stars indicate feature values, and the short vertical lines indicate the threshold candidates. As shown in Figure

5.3, the threshold candidates are in the middle of the two adjacent feature values. For each candidate, the accumulated error of Potsu weak learner h

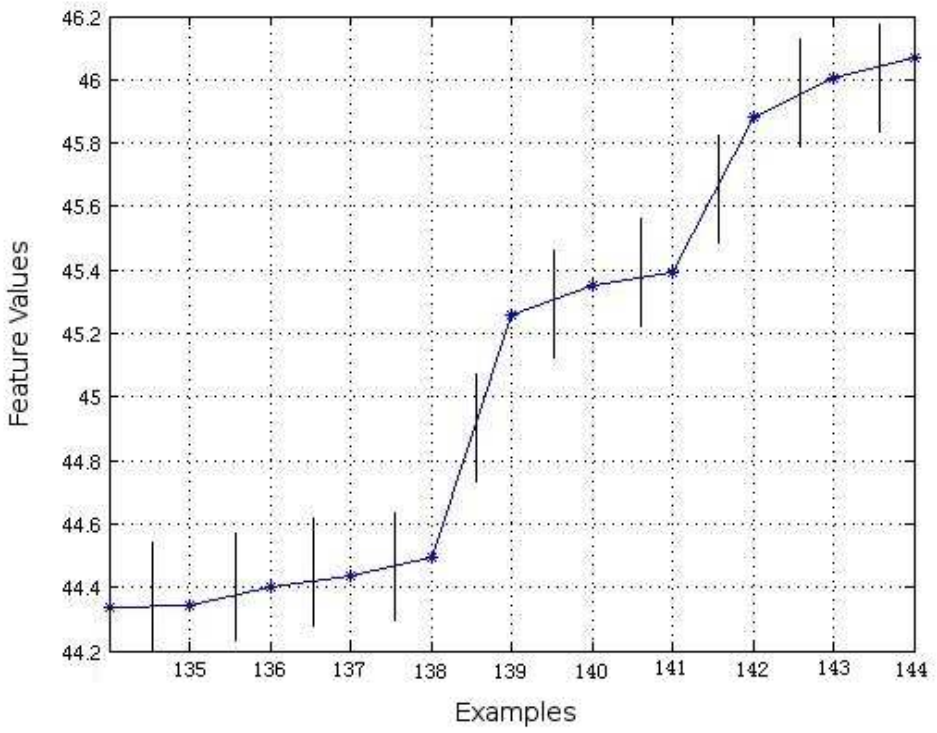


Figure 5.3: Threshold candidates and feature values

with the threshold θ_k is calculated by

$$\varepsilon_k = \sum_{i=1}^{n} \omega_i |h(x_i) - y_i|^2 \tag{5.5}$$

Once ε_k is greater than 0.5, the calculation on the accumulated error is ~~ceased to further examples,~~ and the candidate is no longer taken into account in the following selection. This is because, the weak learner should perform better than a random guess [47] for preventing the loss of any generality.

5.1.4 Summary on Potsu Weak Learner

The Potsu weak learner adopts a heuristic approach to find an optimal threshold reasonably close to the best possible solution. With a large number of examples, the accuracy on predicting the threshold is very high, due to a large number of candidates in selection. In addition, finding an optimal threshold in the Potsu weak learner is very fast due to the perceptron prototype, because Perceptrons are especially suited for simple problems in pattern classification. The classification in perceptron is based on thresholding, which is one of fastest algorithms for segmentation and recognition in pattern classification. The Potsu weak learners adopt a threshold selection technique. In this technique, the algorithm firstly is to collect a set of threshold candidates, and then evaluating the training performance on each candidate, and finally the candidate with the minimal error is selected as the optimal threshold. In [50, 95, 143], weak learners are modelled as the log likelihood ratio (LLR) (see Appendix D), which requires to predict probability density functions. Obviously, Potsu weak learner is faster than those LLR based weak learners which demand the complicated prediction on the probability density function.

The Potsu weak learners are fast in processing one-dimensional data, but it could be exceedingly computationally expensive in higher dimensional data. In one-dimensional data, the number of threshold candidates is $n - 1$ from n examples. However, if the data is expanded into a 2-D space, the number of candidates will be increased to $(n - 1)^2$. If the data is in an m-D data, the number will be $(n - 1)^m$. It is concluded that the number of candidates is increasing exponentially with the number of dimensionality of the data. More candidates in training a Potsu weak learner lead to higher computational cost. Therefore, the Potsu weak learners are most suitable in low-dimensional data.

5.2 Feature Selection by Potsu

The AdaBoost training is used to select significant features from a large feature set. Since AdaBoost training is ~~notorious for a time consuming process~~, optimisation techniques are proposed to speed the process up. The optimisation techniques and the Potsu weak learners are integrated into AdaBoost, and a new variant of AdaBoost - Binary Gabor Boosting algorithm is presented. In the experiments, the results from selected features are discussed with three different feature pre-selection schemes, *i.e.*, the **Full** scheme, the **Interlace** scheme, and the **Scaling** scheme.

5.2.1 Optimisation on AdaBoost

~~Same as~~ discussed in Section 4.4.3, each iteration in Binary Gabor-Boosting training rejects some features which have higher error rate. This is used to reduce the computational cost meaningfully during the training. The reason is that weak learners with high error can not be integrated into a strong classifier. The following considerations are taken.

- ~~Negative examples are largely more~~ than positive examples.
- Positive examples are more important than negative examples.
- The performance on positive and negative examples can be evaluated separately.

In the XM2VTS experiment, the ratio between the number of positive examples and negative examples is highly unbalanced due to the nature of the problem and database. If a strong classifier is trained based on these unbalanced data, the ability to recognise negative examples will be much stronger than the ability to recognise positive examples. However, in a face verification system, the ability to recognise a client should be the most important concern, because the purpose of such system is not to recognise impostors. The positive examples are more important than the negative examples. At the beginning of the training, the positive and negative examples are given different weights according to the number of examples in training. Hence, the

performance of positive and negative examples can be evaluated separately. It indicates that rejection or acceptance of a weak learner will be not only based on its overall classification error, but also on negative examples' error with respect to the weights and false positive rate.

Based on these considerations, the Binary Gabor-Boosting algorithm rejects some "unqualified" features in each iteration. In the XM2VTS experiment, the rejection on a Potsu weak learner is based on negative examples' error with respect to the weights $\varepsilon_{y_i=0}$ and false positive rate $\mathcal{FP}$. The error $\varepsilon_{y_i=0}$ should not exceed a half of the sum of weights belonging to negative examples as

$$\varepsilon_{y_i=0} \leq \frac{1}{2} \sum_{i=1}^{n} \omega_i (1 - y_i) \tag{5.6}$$

For any $\varepsilon_{y_i=0}$ disobeys Equation 5.6, the corresponding feature will be removed from the training set. The term $\sum_{i=1}^{n} \omega_i (1 - y_i)$ is the sum of weights of all negative examples. If all negative examples are misclassified and all positive examples are classified correctly, the error ε will be equal to $\sum_{i=1}^{n} \omega_i (1 - y_i)$. The half of the sum means that the weak learner has just achieved classification performance equivalent to a random guess. At the beginning of AdaBoost training, $\frac{1}{2} \sum_{i=1}^{n} \omega_i (1 - y_i)$ is equal to 0.25. After several iterations, the number is changed, due to updating the weight on each example. To build a strong AdaBoost classifier, weak learners should have ~~the~~ classification performance better than a random guess over all negative examples.

The false positive rate $\mathcal{FP}$ is a term to describe a statistical error of mis-classification on positive examples. Due to only four positive examples in the training set, the possible $\mathcal{FP}$ could be 0.0, 0.25, 0.5, 0.75 and 1.0 corresponding to 0, 1, 2, 3, 4 positive examples misclassified. In the Binary Gabor-Boosting algorithm, the $\mathcal{FP}$ of each weak learner should not exceed 0.5, *i.e.*,

$$\mathcal{FP} < 0.5 \tag{5.7}$$

which means it is not allowed more than 2 positive examples mis-classified.

See separate comment on covering note about false positive definition.

By controlling $\varepsilon_{y_i=0}$ and $\mathcal{FP}$, the number of features is dramatically reduced ~~from~~ each iteration during the training. This greatly reduces the computational time. ~~Number~~ of features trained in each iteration from the first client to the eighth client is displayed in Figure 5.4. The pre-selection scheme is the **Interlace**, so that there are 29120 features at the beginning of the training. From Figure 5.4, after the first iteration, roughly a half of features are removed from the ~~whole set of features.~~ In the eight clients, six clients have less than 15000 features at the second iteration, and two clients (the second and the eighth) have slightly more than 15000 features. The number of features drops significantly when the iteration goes further. From Figure 5.4, from the first client to the seventh client, the number of features in the seventh iteration ~~is dropped~~ below 1000. It was planned in the training to select 200 features. The computational optimisation is used to converge the training in 30 to 70 iterations.

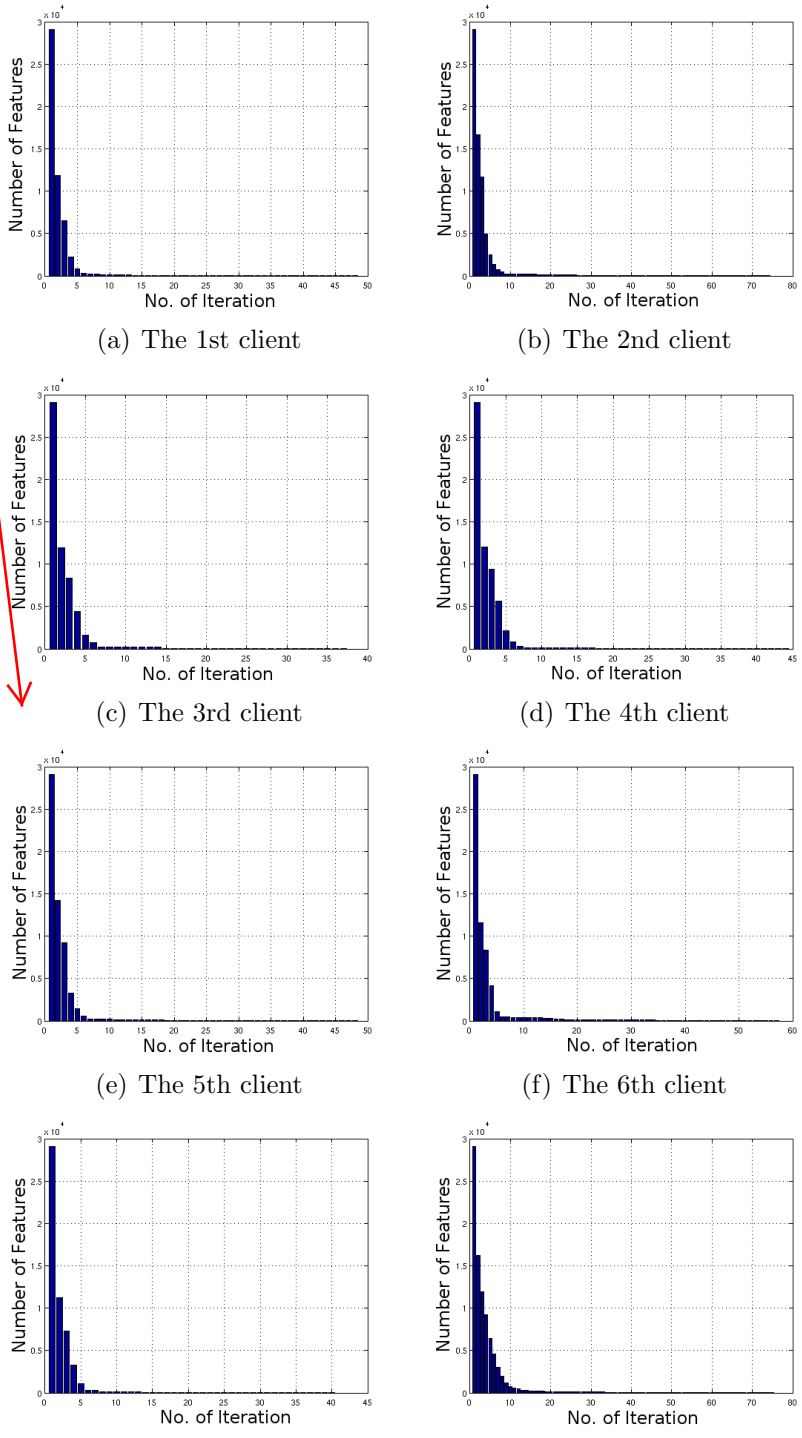
Don't follow this.

Table 5.2 shows the difference on the total number of weak learners trained ~~between with the optimisation~~ and without the optimisation, and the ratio on the time cost. Consequently, the time ratio between the optimisation and no optimisation ~~is varied~~ from $1:18$ to $1:30$, which proves that the optimisation ~~has reduced~~ the computational cost of training significantly. For example, training a feature on an *Intel Pentium 4* 2.8 GHz machine takes 70 milliseconds. For the first client, the training without optimisation will take $1,396,632 \times 70\text{ms} \approx 97,764\text{s}$, *i.e.*, roughly 27 hours; however, the training with the optimisation will take $53,302 \times 70\text{ms} \approx 3,731\text{s}$, *i.e.*, roughly one hour.

Another time-reducing optimisation technique, as described in Section 4.4.3, is to reject any positive or negative ~~feature which has the~~ error greater than 0.5. This optimisation method is named ~~as~~ *optimisation 1*, and the one evaluating positive and negative examples separately is called *optimisation 2*. The *optimisation 1* can save the computational cost around 50% [183]. The comparison chart between the training without any optimisation, with *optimisation 1*, and with *optimisation 2* is given in Figure.5.5. The Figure shows the number of remaining features in each iteration ~~from total~~ 20 iterations. Because there are different specifications on the training images,

The Figure is much too far away. Move up figure 5.4 and table 5.2 ahead of this paragraph .

The large drop after the first iteration makes it difficult to see what is happening on later iterations.



(a) The 1st client (b) The 2nd client

(c) The 3rd client (d) The 4th client

(e) The 5th client (f) The 6th client

(g) The 7th client (h) The 8th client

Figure 5.4: The number of features in each iteration for 8 clients

Table 5.2: The difference between the number of iterations with and without the optimisation

Client	Number of Iterations	Total number of weak learners trained (with optimisation)	Total number of weak learners trained (without optimisation)	Ratio (with:without)
C1	48	53302	1396632	1:26
C2	74	72088	2152105	1:29
C3	37	58663	1076774	1:18
C4	44	61451	1280334	1:20
C5	48	60920	1396632	1:22
C6	57	61592	1658244	1:26
C7	40	54249	1164020	1:21
C8	75	90460	2181225	1:24

the beginning number of features in *optimisation 2* is more than the number in *optimisation 1*. After the training of the first iteration, the number of remained features in *optimisation 2* quickly drops to half of the original number, but in *optimisation 1*, there is only a slight drop on the number of features. After the fifth iteration, the remaining features in *optimisation 2* can be ignored because it is very small compared with the number in *optimisation 1*. Therefore, the optimisation method - *optimisation 2* is the most efficient for reducing the computational cost.

5.2.2 Binary Gabor Boosting algorithm

italics

A new AdaBoost algorithm, named ~~as~~ Binary Gabor-Boosting algorithm, is proposed with the Potsu weak learners and the optimisation 2. The new AdaBoost algorithm is designed not only to select significant features, but also to construct the strong AdaBoost classifier. The strong classifier is composed of several Potsu weak learners which are built on the corresponding Gabor wavelet features. It is described in Table 5.3 -(impostor: negative examples; client:positive examples).

The experiment is operated on two notable face databases - XM2VTS and FERET. In this section, the results from XM2VTS are discussed, while

Table 5.3: Binary Gabor Boosting algorithm for feature selection and classification

1: Given example $(x_1, y_1), \ldots, (x_n, y_n)$, where x_i is the data of the ith example, which are contributed by k features $\{j_1, \ldots, j_k\}$, 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** $t = 1, \ldots, T$ **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** $\{j_1, \ldots, j_k\}$ **do**
6: Train a Potsu weak learner h_j built with one single feature j with the weights $\omega_{t,i}$.
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 of the weak learner h_t as a significant feature.
11: Remove the feature j_t from the feature set $\{j_1, \ldots, j_k\}$.
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: Choose the importance $\alpha_t = \log \frac{1}{\beta_t}$, and $\beta_t = \frac{\varepsilon_t}{1-\varepsilon_t}$.
14: **end for**
15: The final "strong" (ensemble) classifier

$$H(x) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ \\ 0 & \text{otherwise} \end{cases}$$

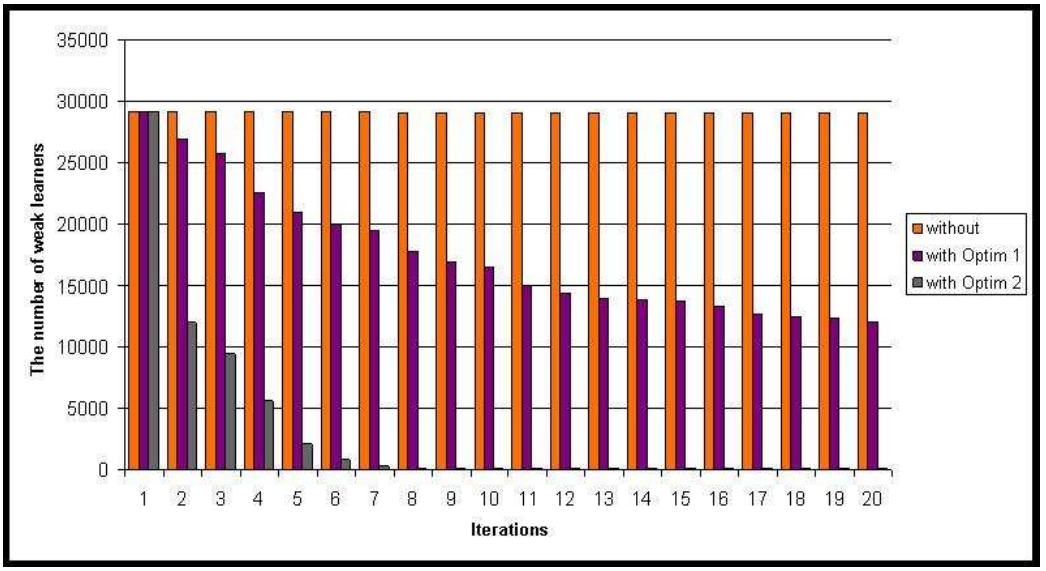


Figure 5.5: The training time comparison between no optimisation, optimisation 1, and optimisation 2

the results from FERET will be discussed in Section 5.7.

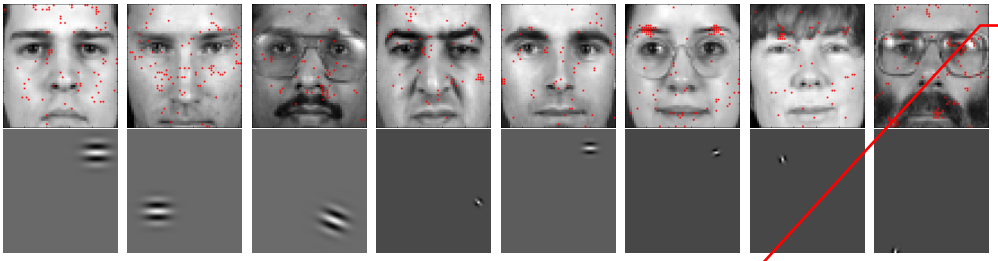
5.2.3 Result discussions

The Binary Gabor-Boosting algorithm is trained and tested with the XM2VTS face database. The algorithm is based on each particular client, *i.e.*, subject, so that the images belonging to a client are treated as positive examples, and others are negative examples. In the first scenario, due to 8 images for each client and 200 clients available, the first four images (named as No.1, No.2, No.3 and No.4) of each client are collected into the training set. There are 800 examples in the training set, and 800 in the testing set. Given a particular client to verify, the positive examples are the first four images of the client, and the negative examples are other 796 images. Feature pre-selection schemes also are applied to remove non-useful features for reducing the computational cost before running the feature selection algorithm. The comparison of the **Full size** scheme, the **Interlace** scheme and the **Scaling** scheme is given here.

Features with the full-size pre-selection scheme

After the Gabor-Boosting training, a small set of features are selected as significant features for each client. The number of selected features from each client is different. From the 1st client to the 8th client, there are 75, 103, 54, 65, 60, 110, 52, 109 features, respectively. As the Full pre-selection scheme does not reduce the original number of features, the original feature set is larger than the ones in the Interlace and Scaling schemes.

Figure 5.6 displays the selected features projected on the corresponding client face image in the top row and the first Gabor wavelet features for each client in the bottom row. For most clients, these selected features are ran-



You used a different word for mole in earlier chapters. I think "mole" is better but be consistent.

Figure 5.6: The selected features on face images and the first selected Gabor wavelet feature with the **Full** feature pre-selection scheme.

domly distributed on faces. These features are also distributed on common facial feature areas, such as eyes, eyebrows, nose, and mouth. Interestingly, there are features located on the mole of the 4th client's face. For the 6th client, two clusters of features are located around the two eyebrows. Similar with the 7th client, some features also cluster around eyebrows. Feature clusters ~~are appeared~~ in the 8th client face image.

Features with Interlace pre-selection scheme

Each client has different number of features selected, ~~and they are varied from 40s to 70s.~~ Figure 5.7 displays the selected features projected on the corresponding client face image in the top row and the first Gabor wavelet feature for each client in the bottom row. These selected features are randomly distributed across faces. It is observed that feature clusters disap-

Why is this interesting - I'd have thought it was obvious?

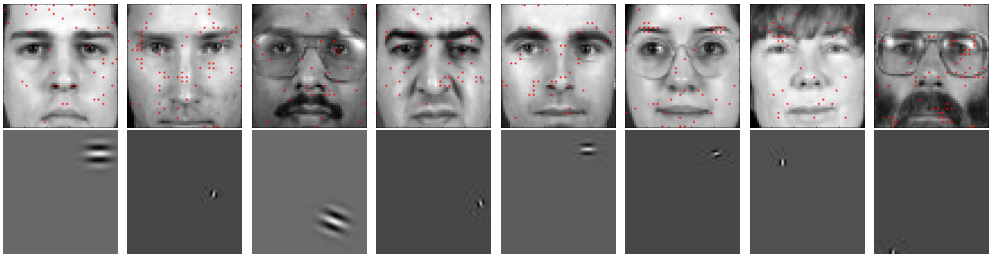


Figure 5.7: The selected features on face images and the first selected Gabor wavelet feature with the **Interlace** feature pre-selection scheme.

pear~~ed~~. Common facial feature areas, such as eyes, eyebrows, nose, and mouth, attract related features. Interestingly, there are some features located on nostrils both in the first client and fifth client. It may indicate that ~~the nostrils of them~~ could be an important feature for recognition. For the second client, more features are located around the eyes and the nose. For the third and eighth clients who wear glasses and have mustache, there are more features around ~~the~~ two areas. The fourth client shows more features around the eyes, the eyebrows, and the gap between cheek and nose. The sixth client who also wears glasses, however ~~more features are~~ located along ~~with~~ the eyebrows rather than the glasses. In the seventh client, the person is different from others, because she have showed her hairs on the forehead ~~which covers~~ the eyebrows. Therefore, some features are laid on the plausible eyebrows area.

Features with the Scaling pre-selection scheme

The Scaling pre-selection scheme scales the response images down in higher spatial frequency. For a feature with higher spatial frequency, it is meaningless to ~~be projected~~ the feature onto the original face image, since the coordinate has been changed. To display the feature correctly, it is necessary to project the feature onto the scaled face image. Figure 5.8 shows the projected features and the first Gabor wavelet feature on each client. The first row shows the original images, and the red dots represent the features with the spatial frequency $\nu = -1$. The second, third, fourth, and fifth rows are the face images which have been scaled down, and those red dots on images

Figure 5.8: The selected features on face images and the first selected Gabor wavelet feature with the **Scaling** feature pre-selection scheme.

represent the selected features of the spatial frequency $\nu = 0, 1, 2, 3$, respectively. Most selected features are with the lowest spatial frequency $\nu = -1$. There are a few features with higher spatial frequency $\nu = 0, 1, 2$, as shown in Figure 5.8. No features at the spatial frequency $\nu = 3$ is selected in all eight clients. The number of selected features from each client is different. From the 1st client to the 8th client, there are 60, 45, 39, 30, 52, 83, 30, 62 selected features, respectively. It is observed that the number of selected features is approximately same as the number selected by the **Interlace** scheme, and smaller than that with the **Full** pre-selection scheme.

5.2.4 Summary on Feature Selection

In this section, the feature selection by using Potsu weak learners is investigated. Similar to the FLD weak learner based AdaBoost feature selection,

some optimisation techniques have been applied to reduce the computational time. The *Optimisation* 2 is different from the *Optimisation* 1 in Chapter 4, and the training with the *Optimisation* 2 is much faster than with the *Optimisation* 1. With the Potsu weak learner and the *Optimisation* 2, the Binary Gabor Boosting algorithm is proposed. The results of selected features are displayed and discussed. With three feature pre-selection schemes, there are some observations from selected features. The number of selected features is different with different schemes. The features are randomly distributed on the faces. Some selected features are located on common facial landmarks.

5.3 Ensemble classification by AdaBoost

In the XM2VTS face database, the features are selected and the strong classifier is built for each client. The experiment ~~has been carried~~ on eight clients separately, and a small group of features are selected for each client. For each client, the first four images are taken for training, and the other four images are for testing. For 200 clients, the training set has 800 face images, and the testing set has 1400 face images[2]. Given a specific client, four images ~~are belonging~~ to the client, and the ~~rest~~ 796 images are ~~taken as the~~ impostors, so ~~that~~ the extremely unbalanced ratio between the number of positive and negative examples is $4 : 796$.

The classification results of the ensemble classifier H from the first client to the eighth client are shown in Table 5.4. All ensemble classifiers have the best performance on the training set, where all classifiers have achieved zero error, *i.e.*, ~~none~~ mis-classification. Both false positive $\mathcal{FP}$ and the false negative $\mathcal{FN}$ are equal to zero so ~~that~~ the ensemble classifiers have achieved the perfect results in the training set. However, in the testing set, the ensemble classifiers reveal poor classification performance on positive examples. All positive examples are misclassified except the 5th client, but all negative examples are classified correctly.

[2]200 clients with 4 images and 95 impostors with 8 images

Don't see how this makes 1400 images unless it should be 75 impostors.

Table 5.4: The classification performance of AdaBoost Classifier

Client No	Number of Features	Training Set (%)			Testing Set (%)		
		Error	$\mathcal{FP}$	$\mathcal{FN}$	Error	$\mathcal{FP}$	$\mathcal{FN}$
1	48	0.0	0.0	0.0	0.5	100	0.0
2	74	0.0	0.0	0.0	0.5	100	0.0
3	37	0.0	0.0	0.0	0.5	100	0.0
4	44	0.0	0.0	0.0	0.5	100	0.0
5	48	0.0	0.0	0.0	0.5	75	0.0
6	57	0.0	0.0	0.0	0.5	100	0.0
7	40	0.0	0.0	0.0	0.5	100	0.0
8	75	0.0	0.0	0.0	0.5	100	0.0

5.4 Overfitting

Experimental results in Section 5.3 suggest the performance of the ensemble classifier on the training set does not represent its overall performance. The ensemble classifier is trained to perform perfectly on the training set, but reveals poor performance on other data sets, *e.g.*, the testing set. This phenomenon is so called *overfitting*. The overfitting is a wide research topic in pattern recognition, machine learning, data mining, etc. There are many issues that lead to overfitting. In this case, the issues causing the overfitting are categorised into two main sources: inferior training data and classification algorithm.

5.4.1 Overfitting caused by the training data

The inferior training data may lead to overfitting, which refers to the problem that classifiers are too specific, or too sensitive to the particulars of the training dataset used to build the classifier. Inside this source, there are two issues that lead to model overfitting, and they are **Curse of dimensionality** and **Selection bias**.

Curse of dimensionality

The overfitting in this case is partly due to small number of positive examples in the training set. Generally, the number of training examples is related to the number of features used in classification. In [75], it is stated that the number of training examples per class should be at least ten times ~~as~~ the number of features. If the number of training examples does not satisfy ~~the~~ requirement, the performance of classification will be degraded. This phenomenon is called "curse of dimensionality". The number of features used in the classification is around ~~from~~ 30 to 70. According to the requirement in [75], the number of training examples per class should be around ~~from~~ 400 to 700. In this case, the number of training examples of positive class is only four, which is far less than the required number. The performance in the testing set on positive examples is very poor, since the discriminating power on positive examples suffers from the curse of dimensionality. However, there are 796 negative examples given in the training set, which are more than ~~as~~ ten times ~~as~~ the number of selected features. The curse of dimensionality is avoided on the negative examples, so that the discriminating power on the negative examples is very strong, *i.e.*, zero false negative rate. In this case, the reason of overfitting is the number of positive examples is far less than the number of selected features, which causes the curse of dimensionality, and the curse of dimensionality leads to the overfitting phenomenon.

Selection bias

Since the standard face database is used in the experiment, the total number of training examples is fixed. With a small number of positive examples and a large number of negative examples, the biased training data ~~implys~~ *Selection bias* [166]. The selection bias possibly happens under some circumstance, in which one kind of examples may preferentially be more included, but other kinds of examples may be more excluded in the training set. ~~The circumstance~~ causes the measurement of statistical significance, *i.e.*, the discriminating power, to appear much stronger or weaker than other kinds of examples. More critically, it may possibly cause completely deceptive re-

Very convoluted - Sentence needs rewriting to make sense.

sults. In this case, the ratio between positive and negative examples are $4 : 796$, so that the discriminating power on negative examples is very strong ($\mathcal{FN} = 0$), but the discriminating power on positive examples is very weak ($\mathcal{FP} = 100\%$). Hence, the selection bias in training dataset affects the results of classification.

5.4.2 Overfitting caused by classification algorithm

A phenomenon that AdaBoost prone to overfit has been observed in [61, 111]. In [120], it concludes that ensemble classifiers in AdaBoost may suffer from overfitting in the presence of noise which may explain some of the decreases in performance. Inside the source, there are two issues pertinent to overfitting. The first issue is that the AdaBoost training over-emphasises noisy examples, and the second issue is due to the weighted voting used by ensemble classifiers.

Over-emphasis on noisy examples

In terms of measurement, a training dataset contains some level of noise. Freund and Schapire [46] suggested that the poor performance of AdaBoost ensemble classifier results from overfitting the training dataset since later training may be over emphasising examples which are actually noise. In AdaBoost, an example with a higher weight is more emphasised in the training than other examples with lower weights. After each iteration of AdaBoost, the weights of some mis-classified examples are increased. In the next iteration, the training will more emphasise on these examples which are often referred to "hard" examples. Because the AdaBoost training always ~~focus~~ on "hard" examples, a "strong" ensemble classifier is constructed at the end. However, if these "hard" examples include high level noise (or actually noise), the AdaBoost training ~~becomes learning~~ the noise rather than the nature of data itself. For a problem with noise, focusing on misclassified examples may cause the ensemble classifier to focus on boosting the margins of noise, but not on boosting the margins of examples. In fact, the weight updating misleads the AdaBoost training in overall classification. In [120], noises from different level are added into data sets, and the results demon-

strate the error rate of the AdaBoost may indeed increase as the number of weak learners increases. In [141], it suggests that later weak learners focus on increasing the margins for examples. Early stop is useful in AdaBoost training for the purpose of increasing performance. In general, AdaBoost ensemble's poor performance may be partially explained by overfitting noise.

Weighted voting

Within the ensemble classifier, each weak learner is associated with an importance α_t, which ~~also is~~ considered as a weight ~~to each~~ weak learner. Hence, the output of ensemble classifier is the combining weighted voting from all weak learners. Previous work [151] has shown how different weight voting scheme can generally be resilient or lead to overfitting. In the work, optimising the weights is the method of choice for achieving minimal generalisation error. In general, uniformly distributed weights may cause the overfitting, while with optimised weights the ensemble classification is insensitive to overfitting.

5.5 Solutions to Overfitting

Two main sources leading to overfitting in AdaBoost ensemble classifiers have already been discussed . This section ~~is to use~~ some techniques to suppress~~ing~~ each source respectively, and to examine the performance of classification. For the inferior training data, the training data is reorganised by applying cross validation, and the real performance of algorithm is revealed. To investigate whether AdaBoost is prone to overfitting in this case, an SVM classifier is adopted with respect to the selected features, and the overfitting effects are suppressed, and a better generalisation on performance is demonstrated.

5.5.1 Cross Validation

The overfitting caused by inferior training data is due to two reasons - curse of dimensionality and selection bias. Both issues ~~are arisen~~ from the very small number of positive examples. To reduce the level of curse of dimensionality and selection bias, more positive examples should be used in the training

dataset. However the number of positive examples is fixed in the database. Hence, cross-validation technique is used to rotate the positive training and testing examples, and help revealing the true performance of algorithm.

Cross-validation [86, 34], also called rotation estimation, is the statistical approach to divide a dataset into two subsets. The analysis is initially performed on one subset, while another subset is retained for subsequent use, such as confirming and validating the initial analysis. The initial subset is the training set, and another subset is the testing set, also called validation set. The performance can be estimated by averaging over all possible divisions of subsets.

In this section, a *leave-one-out* cross-validation has been used. As the name suggests, the leave-one-out cross-validation uses a single example of each class from the original dataset as the testing dataset, and the remaining examples for training. It is repeated in turn such that each example of the class in the dataset is used once as the testing data. Errors are estimated by simply averaging over the turns. The leave-one-out cross-validation is an excellent tool to test the robustness of a classifier, which is sensitive to small changes in the training set. If a classifier performs well under the leave-one-out cross validation, it suggests that the classifier is satisfactory.

There are eight face images for each client in the XM2VTS database. By using the leave-one-out cross-validation, the training set and the testing set have been changed to seven images for training and one image for testing. To a particular client, the number of examples in the training set has been increased to 803, which involves 7 positive examples and 796 negative examples. The leave-one-out cross validation is carried out 8 times. Table 5.5 shows the performance on the 4th client with the **Scaling** pre-selection scheme. As indicated in Table 5.5, validations work very well except of that the cross-validation leaves the 4th image out. In the cross-validation, the only one positive example in the testing set is mis-classified, but all other negative examples are classified correctly.

By applying *leave-one-out* cross-validation, the number of positive examples is increased, so that the degree of cures of dimensionality and selection bias are reduced in the case. In the original training set, the number of

Table 5.5: Performance of leave-one-out cross-validation on the 4th client

Leave-one-out	Number of Features	Training Set (%)		Testing Set (%)	
		$\mathcal{FP}$	$\mathcal{FN}$	$\mathcal{FP}$	$\mathcal{FN}$
1	38	0.0	0.0	0.0	0.0
2	28	0.0	0.0	0.0	0.0
3	19	0.0	0.0	0.0	0.0
4	39	0.0	0.0	100	0.0
5	18	0.0	0.0	0.0	0.0
6	37	0.0	0.0	0.0	0.0
7	40	0.0	0.0	0.0	0.0
8	36	0.0	0.0	0.0	0.0

positive examples is four, while in these cross-validation sets, the number becomes seven. Although the number of positive examples is still much less than the number of features (around from ~~40s to 70s~~) and the selection still bias to negative class, the overall performance of classification is significantly increased. In general training, to suppress overfitting, the training data should have the number of examples per class ten times more than the desirable number of features, and the number of examples in each class should be roughly equal. However, in this case, with small increase in the number of positive examples, the performance has improved greatly.

5.5.2 SVM

Since AdaBoost prone to the overfitting, an alternative classifier - SVM is used to classify the examples. In this section, an SVM classifier is trained with the given selected features on the training set.

Results from SVM classification

With eight clients, eight SVM classifiers are trained with a polynomial kernel with degree of three. From the 1st client to the 8th client, there are 48, 74, 37, 44, 48, 57, 40 and 75 features used to construct the SVM classifiers. The training of the SVM actually is to adjust infrastructure between each

selected features and maximise the margin between the positive and the negative. After the SVM classifiers are well trained, the training set and the testing set are given for the testing purpose. Table 5.6 shows the performance of SVM classification belonging to eight clients. On the 1st, 2nd, 4th, 5th

Table 5.6: The performance of SVM classification on eight clients with all selected features

Client	Number of Features	Training Set (%)		Testing Set (%)	
		$\mathcal{FP}$	$\mathcal{FN}$	$\mathcal{FP}$	$\mathcal{FN}$
1	48	0.0	0.0	0.0	0.0
2	74	0.0	0.0	0.0	0.06
3	37	0.0	0.0	50	0.0
4	44	0.0	0.0	0.0	0.0
5	48	0.0	0.0	0.0	0.0
6	57	0.0	0.0	0.0	0.0
7	40	0.0	0.0	100	0.13
8	75	0.0	0.0	100	0.0

and 6th clients, these SVM classifications have achieved perfect performance, in which all positive examples in both the training set and the testing set are classified correctly. For the 2nd client, only one negative example is recognised as positive, and the false negative rate $\mathcal{FN}$ is 0.06% which is very small and can be neglected. Hence, the SVM can achieve low error rate. However, when the SVM encounters some "hard" clients, the poor performance apparently exists. The 8th client still gets overfitting with the training set and has a big false positive in the testing set. Even more, the 7th client presents a slightly worse performance in which the classifier not only overfits the training set, but also causes exiguously false negative in the testing set.

Suppressing Overfitting

In the last section, two issues for ensemble classifiers in AdaBoost to be overfitted are presented: the issue of over-emphasising on noisy examples and

the issue of optimising weights. From the perspective of these two issues, the SVM is used to suppress the overfitting effects.

Support Vectors In [46], AdaBoost training emphasise on noisy examples at the later iterations, so that the overfitted result is acquired at the end of training. The training of SVM is the optimisation process which is to find the primal form of the Lagrange formalism. By optimising the Lagrange function, a small group of support vectors (SVs) are obtained, which defines the hyperplanes between classes. The SVs are some examples which have their non-zero Lagrange multipliers. In this case, there are 5 to 11 examples taken as SVs from ~~totally~~ 800 training examples. The numbers of SVs in the SVMs based on these eight clients are shown in Table 5.7. Hence, SVMs

Table 5.7: The numbers of SVs in the SVMs based on eight clients

Client	No of SVs	No of Positive SVs	No of Negative SVs
1	7	1	6
2	11	2	9
3	5	1	4
4	10	1	9
5	9	1	8
6	7	1	6
7	7	2	5
8	5	2	3

use a small number of examples to construct classifiers instead of the whole set of examples (including the noisy examples). This optimisation process is equivalent to find some examples, *i.e.*, SVs which contain the lowest level of noise. By only using SVs, noisy examples are excluded in the construction of classifier. The training and classification only emphasise on the examples with the lowest level of noise, so that the overfitting caused by noisy data is reduced to the minimum level. In addition, when SVMs are designed originally, the overfitting issues are taken into account. SVMs avoid overfitting by choosing a specific hyperplane among the many that can separate the data in the feature space. SVMs find the maximum margin hyperplane, which

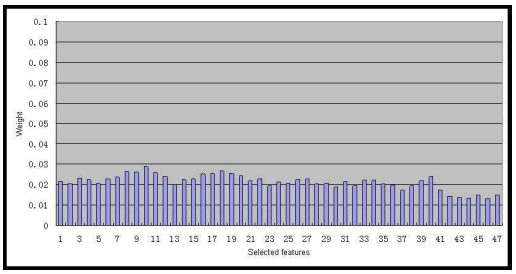
maximises the minimum distance from the hyperplane to the closest SVs

Optimisation on weights Sollich and Krogh [151] displayed that uniformly distributed weights normally lead to overfitting, while with optimised weights the classification is insensitive to overfitting in ensemble classifiers. The ensemble classifier in AdaBoost contains a set of weights for weak learners which vary in a small range. Each selected feature j_t is used to construct a weak learner h_t, and each weak learner is associated with a weight α_t. The weight indicates how important the corresponding feature plays role in classification. When these weights are scaled into a probability distribution, *i.e.*,$\sum \alpha_t = 1$, the scaled weights ~~are showing~~ some similarity to an uniform distribution. The probability distributions of weights from eight clients are shown in Figure 5.9 and 5.10. In the probability distribution, the variances between weights are small, which convey all selected features have roughly equalled importance in classification.

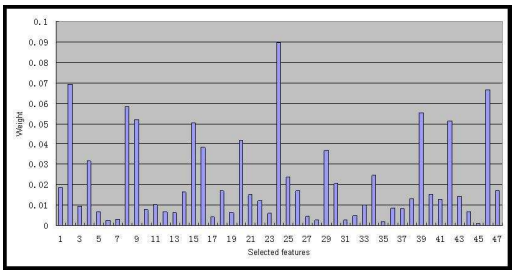
The SVM classifier also contains a set of weights for selected features which are randomly distributed. The training process of SVM is to find a linear discriminant function

$$g(x) = \omega^T \mathbf{x} + b \tag{5.8}$$

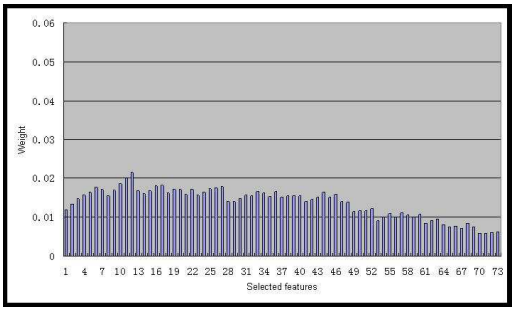
where $\mathbf{x}$ is the vector $\{x_1, x_2, \ldots, x_n\}$ formed by n selected features, ω is the weight vector $\{\omega_1, \omega_2, \ldots, \omega_n\}$ for corresponding n features, and b is a bias. The weights also imply how important the corresponding features play roles in classification. When these weights are scaled into a probability distribution, the weights ~~are varying~~ dramatically. The probability distributions of weights from eight clients are shown in Figure 5.9 and 5.10. ~~A~~ few features have their weights much higher than other features, which indicates only ~~the~~ small number of features play important roles in classification. In this case, the ensemble classifier shows a nearly uniform distribution on weights for selected features, while the SVM classifier shows ~~an~~ optimised weights. Hence, the weights for selected features are optimised, the overfitting effect is reduced in SVMs.



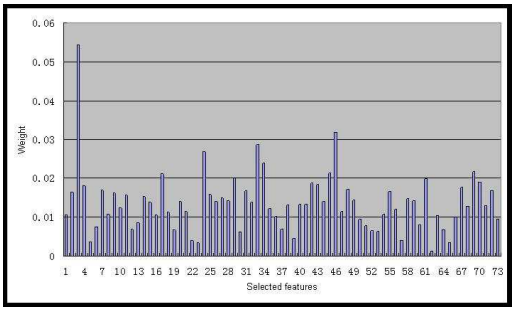
(a) AdaBoost on the 1st client



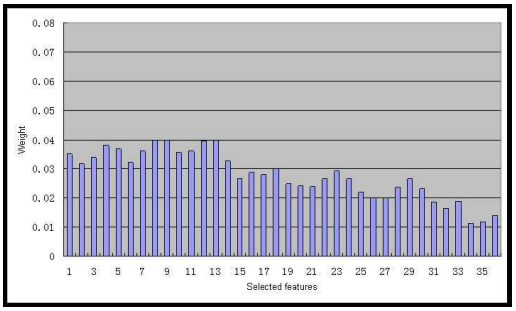
(b) SVM on the 1st client



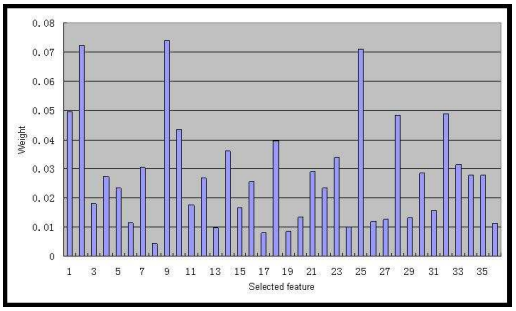
(c) AdaBoost on the 2nd client



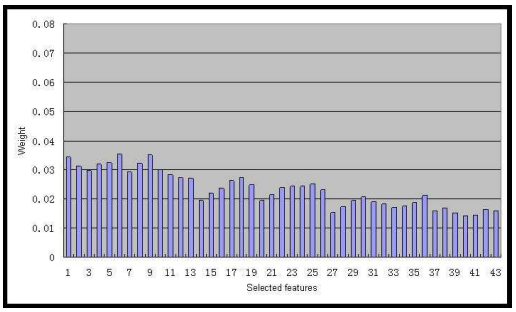
(d) SVM on the 2nd client



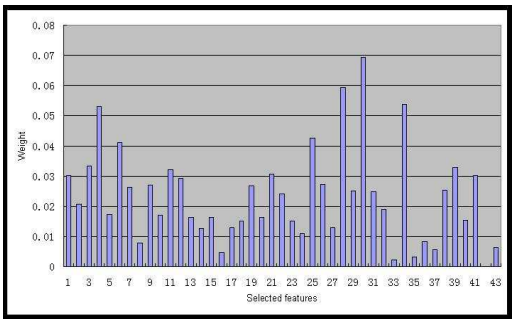
(e) AdaBoost on the 3rd client



(f) SVM on the 3rd client

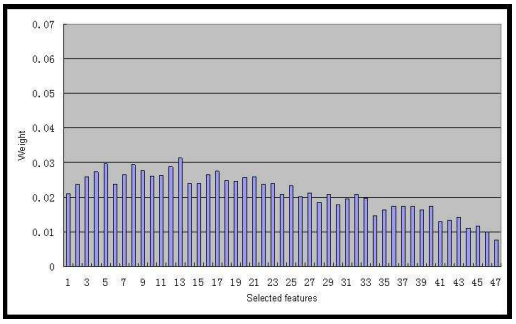


(g) AdaBoost on the 4th client

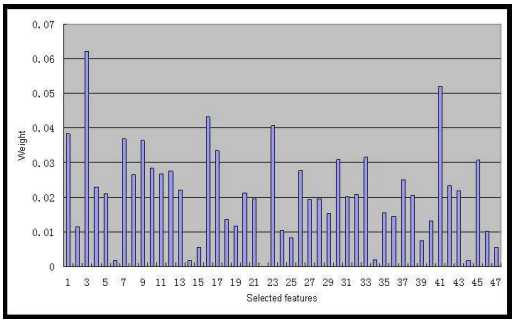


(h) SVM on the 4th client

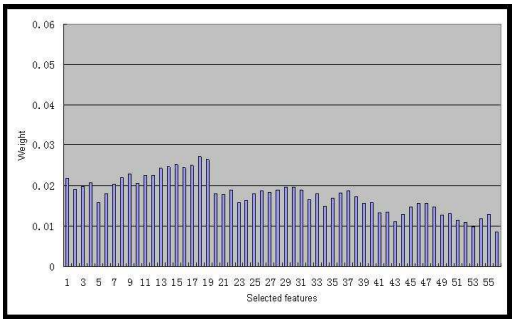
Figure 5.9: The weights' distribution from the 1st client to the 4th client



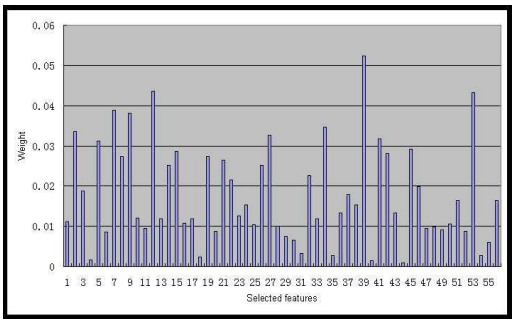
(a) AdaBoost on the 5th client



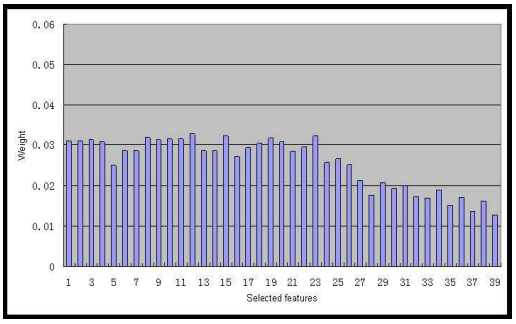
(b) SVM on the 5th client



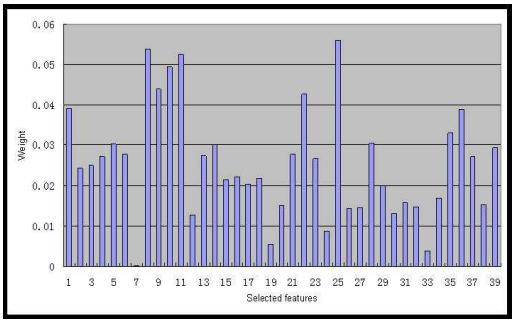
(c) AdaBoost on the 6th client



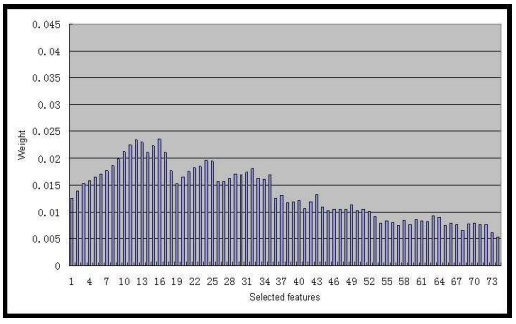
(d) SVM on the 6th client



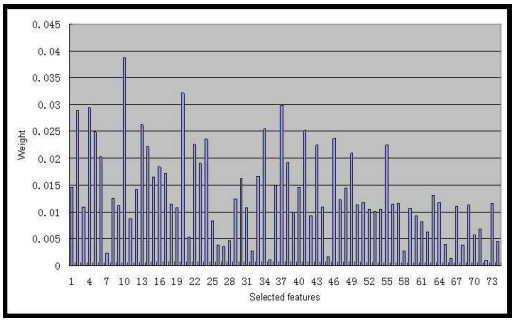
(e) AdaBoost on the 7th client



(f) SVM on the 7th client



(g) AdaBoost on the 8th client



(h) SVM on the 8th client

Figure 5.10: The weights' distribution from the 5th client to the 8th client

The overfitting is suppressed by using SVMs, and the overall performance is improved. Comparing with the results from SVM classification, the ensemble classifier in AdaBoost shows some drawbacks from two issues. Firstly, the ensemble classifiers are prone to fit some noisy examples at later iterations, while the SVMs only focus on some important (non-noisy) examples - SVs. In addition, the ensemble classifiers have a set of nearly uniformly distributed weights, but the SVMs demonstrate some optimisation on weights for improving the performance of classification. Since the SVM classifiers have better overall performance than the ensemble classifier in AdaBoost, the overfitting effects can be partly explained due to the ensemble classifier in AdaBoost prone to overfitting. The results proves that the AdaBoost is indeed susceptible to overfitting in this case, so that it is better to use additional classifiers, such as SVMs, is the classification.

5.5.3 Summary on Overfitting

Both inferior training dataset and ensemble classifier in AdaBoost lead to overfitting. By improving the training dataset or applying alternative classifiers, the overfitting effect is reduced, and even avoided in some cases. When the training dataset is improved by applying cross-validation, the performance is 87.5% hit rate on positive examples and 100% hit rate on negative examples. When the classifier is changed to SVM, the performance of classification has been improved on some clients, which demonstrates using an SVM classifier is an appropriate choice for classification. In the real implementation under a restricted face database, it is impossible to add more new face images into the database, which makes improving the training dataset very difficult. Hence, the only way for suppressing the overfitting effect is to use SVMs as the classifiers. In the proposed algorithm, significant features are selected by AdaBoost algorithm, while the verification is performed by SVM.

5.6 Potsu weak learner v.s. FLD weak learner

Instead of using all selected features, an additional experiment is carried out with only the top 20 features. The evaluation set is fixed in a 20-dimensional feature space for each client. An SVM is trained, and the performance for each client is shown in Table 5.8. For the 1st, 4th, 5th and 6th clients, the

Table 5.8: The performance of SVM evaluation on eight clients with 20 selected features by Potsu

Client	Number of Features	Training Set (%)		Testing Set (%)	
		$\mathcal{FP}$	$\mathcal{FN}$	$\mathcal{FP}$	$\mathcal{FN}$
1	20	0.0	0.0	0.0	0.0
2	20	0.0	0.0	25	0.0
3	20	0.0	0.0	50	0.0
4	20	0.0	0.0	0.0	0.0
5	20	0.0	0.0	0.0	0.0
6	20	0.0	0.0	0.0	0.0
7	20	0.0	0.0	50	0.0
8	20	0.0	0.0	75	0.13

performance is very good with 20 features. For some clients, like the 2nd and 3rd clients, the performance could be improved by adding more features. For the 3rd client, the $\mathcal{FP}$ rate is fixed at 50%, no matter 37 ~~features~~ or 20 features are used. For those "hard" clients, such as the 7th and 8th clients, the performance is varied. Adding more features into the evaluation set does not guarantee the improvement, but leads to overfitting.

Table 5.9 demonstrates the performance comparison of face verification algorithm with Potsu and FLD weak learners. The AdaBoost algorithm is used for feature selection and SVMs for classification. Both SVM classifications ~~are with~~ 20 features. The AdaBoost algorithm with the Potsu weak learner has a better result than the one with FLD weak learners. ~~It~~ indicates that the Potsu weak learner in AdaBoost training contributes to the classification performance compared to the FLD weak learner. In general, the Potsu weak learner is much ~~more~~ superior than FLD weak learners in

Table 5.9: Performance of the Potsu weak learner and the FLD weak learner with only 20 features.

Client No	$\mathcal{FP}$		$\mathcal{FN}$	
	Potsu	FLD	Potsu	FLD
1	0.0	0.0	0.0	0.0
2	0.125	0.25	0.0	0.0
3	0.25	0.25	0.0	0.0
4	0.0	0.375	0.0	0.0
5	0.0	0.25	0.0	0.0
6	0.0	0.25	0.0	0.0
7	0.25	0.5	0.0	0.002
8	0.375	0.5	0.001	0.0

AdaBoost training.

Hyphenation along this edge is horrible - set a higher threshold.

5.7 FERET Testing

The proposed Gabor-Boosting algorithm is also tested in the FERET face database. In this section, the FERET face database is introduced. To accomplish face recognition, pre-processing is applied to FERET face images. By using the Potsu weak learner based AdaBoost, the significant features are selected. In classification, the ensemble classifier and the SVM classifier are performed.

5.7.1 FERET Database

The Facial Recognition Technology (FERET) program and development of the FERET database is sponsored by the DOD (Department of Defence) Counterdrug Technology Development Program through the Defence Advanced Research Products Agency (DARPA) in USA. In addition, the National Institute of Standards and Technology (NIST) ~~is serving~~ as a technical agent for distribution of the FERET database. As a part of the FERET program, the database of face images was collected in 15 sessions between December 1993 and August 1996. The database is used to develop, test and

evaluate face recognition algorithms. In order to display the state-of-art in face recognition technology, the FERET database has been made available to scientists in the research area. The goal of the FERET program is to develop new technology for the automatic recognition on human faces. The program contains three objectives [128].

- To develop algorithms required for a face recognition system.
- To create a large database of facial images for face recognition.
- To test and evaluate the algorithm based on the developed database.

The FERET database contains $1,564$ sets of shot for a total of $14,126$ images with $1,199$ individuals. The whole sets are divided into two portions, the development portion and the sequestered portion. The development portion of 503 sets is released to public for scientific research, while the sequestered portion is reserved by the US government for independent evaluation. The original FERET database was released in 2001 and consisted of $14,051$ gray scale images. The original FERET database is also called the Grey FERET database. In 2003, a colour version of FERET database was released, which contains $11,338$ facial images from 994 individuals. The new FERET database is called Colour FERET database. In the thesis, without mentioning either Colour or Grey, the FERET database ~~is referred~~ to the Colour ~~FERET~~.

Images from the FERET database are 512×768 pixels, and are stored in PPM (Portable Pixel Map) image format. The images have not undergone any lossy compression or processing. The filenames of these images are in the form of *nnnnn_yymmdd_xx_c*. The filename is divided into four parts. *nnnnn* is a five digit integer that identifies the subject (person). These five digit integers are called the ID of a subject. *yymmdd* represents the date when the face image was actually ~~been~~ captured. *xx* shows a pose of the face. There are 13 different poses shown in ~~the~~ Table 5.10. The set *fa* represents the regular frontal image. The set *fb* represents alternative frontal image, which was taken shortly after the corresponding *fa* image was taken. A regular frontal image was captured for every subject at every shot session. In the

Table 5.10: 13 poses in FERET Database with angles, number of images and the number of subjects

Pose	Angle	Images (#)	Subjects (#)
fa	0	1364	994
fb	0	1358	993
pl	+90	1312	960
hl	+67.5	1267	917
ql	+22.5	761	501
pr	−90	1363	994
hr	−67.5	1320	953
qr	−22.5	761	501
ra	+45	321	261
rb	+15	321	261
rc	−15	610	423
rd	−45	290	236
re	−75	290	236

FERET database, each subject includes 5 to 11 images. In each subject, there are at least two front views (*fa* and *fb*). A different facial expression is requested for the second frontal image.

The database also contains ground truth files for the subjects and the images. Ground truths are recorded in two formats - XML file and plain text file. These files give the detailed ground truth of corresponding images, such as gender, with or without glasses, beard, etc. The information of these ground truth files are required in the pre-processing of face images.

5.7.2 Pre-processing on the FERET

In ~~the~~ thesis, only frontal view face images are considered, such that only the *fa* and the *fb* sets are used. As the XM2VTS database, segmentation and pre-processing are also needed to remove useless substances, *e.g.*, the background with shadow, the subject's hair, face, ears, neck, clothes and so on. The original image is segmented into a smaller size of 128 × 128 pixels, which only contains the inner face part. The segmentation requirements are:

- The subimage ~~is~~ with 128 × 128 pixels

- The distance between the centres of two pupils is 64 pixels
- Two eyes must be horizontal
- The height from the centre of pupil to the bottom is 96 pixels

The requirements are as visualised in Figure 5.11. The steps of the segmen-

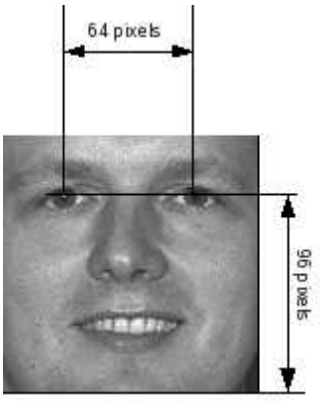


Figure 5.11: The visualisation of the requirements of the subimages.

tation are as follows

1. Calculate the angle θ of two eyes' pupils in the original image I.
2. Rotate the image I with the angle θ, to ~~let~~ two eyes horizontal and generate a new image I'.
3. Calculate the distance d between two pupils (two pupils should be horizontal now).
4. Get a scaling ratio r by $r = 64/d$.
5. Scale the image I' with the scaling ratio r and generate a new image I''.
6. Crop the image I'' based on eyes' coordinates, and get a 128×128 subimage.

By applying the above steps to the original face image, two pupils' coordinates on the original image must be acquired. The coordinates are obtained from the ground truth files. Some images do not have pupils' coordinates provided and a manual detecting program has to be applied. In image rotation, affine transformation [74, 66] is applied on the original image with the rotation matrix to rotate the original image.

Subimages of 128×128 pixels are further reduced to 64×64 pixels with consideration of the computational cost. To prevent information loss, the resize algorithm is adopted with the **Bicubic interpolation** [84], which keeps the best image quality. These processed subimages are shown in Figure 5.12.

Figure 5.12: Samples of Segmented subimages from the FERET Database.

5.7.3 Feature Selection

There are two frontal image sets fa and fb in the FERET database, such that the fa set is used as the training dataset, and the fb set is used as the testing

(a) The training set (11 images)

(b) The testing set (11 images)

Figure 5.13: The 22 face images for the ID00029 subject from the training set and the testing set respectively.

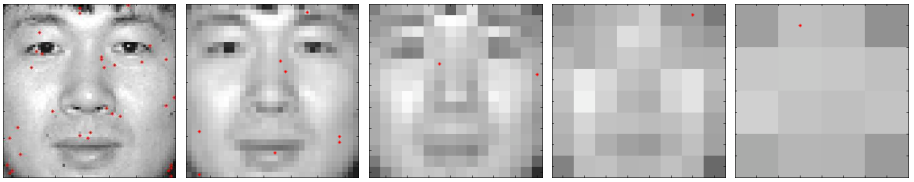
dataset. There are $1,364$ images with 994 different subjects in the training dataset, and $1,358$ images with 993 subjects in the testing dataset. The number of images - examples in both datasets is similar, and the condition of illumination and other environments are unchanged between the fa set and the fb set.

The training and testing on FERET is also the client-based over positive and negative examples. To reduce the degree of curse of dimensionality and selection bias, the subject with ID00029 has been chosen as the client who has 11 images in the training set. There are ~~totally~~ $1,364$ face images, so ~~that~~ the ratio between the positive and negative is 11/1353. ~~Comparing to~~ the XM2VTS, the ratio is still very rare, but there are more positive examples in the training set. In the testing set, the subject with ID00029 also has 11 face images, and the total number of images in the testing set is $1,358$. The 22 face images in the training set and the testing set are shown in Figure 5.13.

After these 64×64 subimages are convolved with 40 Gabor wavelet kernels, the feature space is in the order of $163,840$ dimensions. If the Gabor-Boosting algorithm is directly applied on to ~~these~~ massive amount of features, the computational cost ~~is~~ enormous. Therefore, a feature pre-selection scheme - **Scaling** is chosen. By applying the Scaling pre-selection scheme, the total number of features is reduced to $43,648$.

The Gabor-Boosting algorithm is applied on to the ID00029 subject with the *optimisation 2* in Section 5.2. The training is finished in 45 iterations, so that 45 Gabor wavelet features are selected as the significant features to

verify the ID00029 subject. These 45 features are shown in Figure 5.14. To



(a) $\nu = -1$ (b) $\nu = 0$ (c) $\nu = 1$ (d) $\nu = 2$ (e) $\nu = 3$

Figure 5.14: The selected features from the ID00029 subject in the FERET database are projected on the face image with corresponding scales.

project the features with higher spatial frequency, the face image is scaled according to the frequencies. Most selected features are laid in the frequency $\nu = -1$. These features are around eyes, nose, and mouth, and randomly spread on the faces.

5.7.4 Classification

The performance of classification on the ID00029 subject in the FERET database is tested by the ensemble classifier consisting Potsu weak learners and the SVM classifier.

There are 45 weak learners available for constructing an ensemble classifier with the corresponding importance α_t. The 45 weak learners formed classifier H has achieved 100% accuracy on the training set, in which the false positive and the false negative are both zero. However, in the testing set, 6 out of 11 images for the subject are not recognised. The overall classification error rate is 0.4%, but the false positive rate is 54.5%. Comparing with the results from the XM2VTS, the false positive rate has been reduce and the performance of classification has been improved in the FERET database. The overfitting has been considerably suppressed in the FERET database testing. When the number of positive examples increasing in the training set, the degree of curse of dimensionality is reduce, and the performance of the classification is improved.

However, the false positive rate 54.5% is still rather high. In Section 5.5, since the ensemble classifier is prone to overfitting, the best solution should

be feature selection by AdaBoost algorithm and classification by SVM. So the SVM classifier is used to train the training data with these 45 features and test the performance on the testing dataset. Feeding the 45-dimensional training data into a~~n~~ non-linear SVM with a polynomial kernel, the SVM classifier is well trained. In the testing, the false positive is zero, which means all positive examples are recognised correctly. The false negative only occurs 2 times out of total 1347 negative examples. These two images are 00516_940519_fb and 00588_940307_fb shown in Figure 5.15. Interestingly,

(a) 00516_940519_fb (b) 00588_940307_fb

Figure 5.15: The two false negative occur in the testing dataset.

both individuals are oriental. It may indicate that all oriental faces share similar characteristics on facial appearance. In general, the performance of classifier is greatly improved by using SVMs. Therefore, it proves that the AdaBoost algorithm is appropriate for feature selection, but the SVMs are superior than AdaBoost in classification.

5.7.5 Summary of FERET Testing

In this section, experiments are performed on the FERET face database. To obtain the inner face subimages, pre-processing techniques are applied on the original face images. The significant features are selected by the Potsu weak learner based AdaBoost. With these selected features and corresponding weak learners, an ensemble classifier is constructed. The performance of ensemble classifier on the FERET face database is better than the performance on the XM2VTS face database, but the performance is not ideal.

An SVM classifier is trained on the training set with respect to selected features. The recognition rate of the SVM classifier ~~is achieved~~ 99.99% with only two false negative out of 1347 negative examples. It is concluded that the Gabor-Boosting algorithm combined with an SVM classifier can achieve high accuracy in face verification.

5.8 Summary

In this chapter, a novel weak learner - Potsu weak learner for AdaBoost training is presented. The Potsu weak learner satisfies the requirement of AdaBoost, which demands the minimal error over the training set. The Potsu weak learners are fast due to the simple perceptron prototype, and it is accurate when there is a large number of training examples. In the experiments on the XM2VTS and the FERET face database, significant features are selected by the Potsu weak learner based AdaBoost training. However, the classification made by the ensemble classifiers in AdaBoost illustrates poor performance which is explained as overfitting phenomenon. The overfitting is caused by two sources of factors: inferior training dataset and classifier itself. In the inferior training dataset, the number of positive examples is much less than the number of features in classification, so that it leads to curse of dimensionality and selection bias. In classifier, the AdaBoost is prone to overfitting due to its over-emphasis on noisy data and un-optimised weights. To suppress overfitting, cross-validation is used to improve the training dataset, and SVM classifiers are chosen as the classification solution instead of ensemble classifiers. Both methods illustrate the improvement on the performance of classification. The experiments suggest that AdaBoost is appropriate for feature selection, while classification can be done by using an alternative classifier, such as SVM. Also, the Potsu weak learners based AdaBoost algorithm demonstrates superior performance over the AdaBoost algorithm with FLD weak learners.