Visual Learning Models for Real-world Object Recognition using Diverse Features

野宮, 浩揮

(Degree)
博士 (工学)

(Date of Degree)
2008-09-25

(Date of Publication)
2009-07-31

(Resource Type)
doctoral thesis

(Report Number)
甲4404

(URL)
https://hdl.handle.net/20.500.14094/D1004404

Doctoral Dissertation

Visual Learning Models for Real-world Object Recognition using Diverse Features

（実世界における物体認識のための
多様な特徴を用いた視覚的学習法に関する研究）

August 2008

Graduate School of Science and Technology
Kobe University

Hiroki Nomiya

Contents

List of Figures

List of Tables

Abstract

In recent years, the rapidly improving data processing capacity of computers has made the widespread use of real-world image data in various applications possible despite the large amount of information contained in images. However, given the complex structure of real-world objects in image data, even state-of-the-art computer vision processing frameworks have great difficulty in accurately recognizing various objects. Therefore, a recognition method which can flexibly and precisely discriminate a wide variety of real-world objects is sorely needed. In order to develop such a recognition method, we propose an effective recognition method based on visual learning. The introduction of the idea of machine learning into computer vision processing allows visual learning to perform flexible recognition. By autonomously acquiring knowledge from given image data, visual learning develops the ability to discriminate objects. We propose two types of visual learning methods which are applicable to various recognition tasks. The first method describes and recognizes objects using the features represented by the intensity values of each pixel in an image. Since an image contains a large number of pixels, we propose an efficient feature selection algorithm based on the Set Covering Machine.

In the second method, diverse features such as contour, texture and spatial frequency are used for recognition. To utilize multiple features, we introduce an ensemble learning framework which constructs and integrates multiple visual learners. As a result, our method is able to achieve improved recognition performance compared with existing recognition methods which utilize a single or a few features. In object recognition tasks, however, the recognition performance is negatively impacted by the crucial problem that real-world image data often contain inappropriate images, such as objects which are highly deformed or occluded or are located in cluttered backgrounds. Learning these images results in the degradation of recognition accuracy. Thus, in order to solve this problem, we propose an effective machine learning algorithm called NadaBoost and introduce it into our visual learning.

NadaBoost is based on a widely used ensemble learning algorithm called AdaBoost. One critical drawback of AdaBoost is that its performance is easily worsened by examples that are difficult to classify correctly. In object recognition domains, these examples, called hard examples, correspond to inappropriate images. However, NadaBoost considerably reduces the influence of hard examples by identifying hard examples and determining weight thresholds. By

controlling the weights of hard examples, it prevents the degradation of classification performance which these examples can cause. As a result of the introduction of NadaBoost into visual learning, our visual learning is robust to inappropriate images. Through several object recognition experiments using real-world image data, we verify the effectiveness of our visual learning methods.

Chapter 1

Introduction

As the information processing ability of computers improves, real-world images are more and more widely used in various applications, for example, content-based image retrieval and surveillance system. Thus, flexible and accurate image recognition methods are strongly needed for efficient image data mining. However, real-world images generally contain a wide variety of objects which have complex features (e.g., shapes and textures). Therefore, it is difficult to recognize real-world objects accurately using conventional pattern recognition methods. Most of conventional methods utilize a small number of features while real-world objects are so complex that they cannot be fully described by a few features. Although complex objects can be described using multiple features, finding useful features for precise recognition is difficult because useful features are dependent on objects. Moreover, real-world images often contain considerable noise. These properties make accurate recognition quite difficult. Because of these problems, the recognition performance of current recognition systems for real-world recognition tasks is far from adequate compared with human visual ability.

In order to facilitate the acquisition of a level of recognition ability comparable to that of human visual systems, one effective method consists of introducing learning schemes into image understanding frameworks. Based on this idea, visual learning has been proposed [15]. Visual learning is the learning framework to autonomously acquire knowledge to recognize images (or objects in images) based on machine learning frameworks which analyze given images and construct recognition models to correctly recognize unseen images for given recognition tasks. Visual learning attempts to emulate the ability of human beings that acquires excellent visual recognition ability through observing various objects and identifying several features by which to discriminate them [22]. Similar to the human visual system, a visual learning system first extracts several features from given images. Then, it inputs the features into some kinds of machine learning algorithms to acquire the necessary knowledge. In visual learning, therefore, the learning framework and the strategy to extract, select or construct features are the key points to perform accurate recognition.

As a learning framework to deal with diverse features, we introduce ensemble learning. In ensemble learning, multiple classifiers are constructed using diverse features. Then, the classifiers are integrated into an ensemble classifier whose recognition performance is better than a single classifier. To fully take advantage of ensemble learning, all classifiers cooperate with each other in our ensemble learning in order to optimize their own individual recognition performance. This learning strategy enables more flexible and accurate recognition.

As mentioned above, visual learning makes use of machine learning algorithms for object recognition. Thus, the performance of the machine learning algorithms used in visual learning has a considerable effect on recognition performance. In order to enhance the performance of visual learning, we propose an effective machine learning method based on AdaBoost [11][33]. AdaBoost, a kind of boosting algorithm, adopts an ensemble learning framework [8] in which multiple classifiers are integrated into an ensemble classifier with better classification ability. Thus, this ensemble learning strategy leads to better classification results compared with common machine learning strategies which use a single classifier. However, AdaBoost has one crucial drawback, in that it is easily affected by outliers or noisy data (we call them hard examples). This problem is caused by AdaBoost's algorithm, which gives high weights to hard examples. In order to overcome this drawback, we introduce a weight thresholding algorithm to detect hard examples and reduce their weights. As a result, the proposed boosting algorithm is robust to hard examples. In visual learning, hard examples correspond to objects that are highly deformed or occluded or are found in cluttered scenes. Therefore, applying the proposed method to visual learning, we are able to construct visual learning methods which are fully robust to such objects.

The subsequent sections of this dissertation are organized as follows. Chapter 2 reviews the object recognition studies and machine learning frameworks which are the bases of the proposed visual learning methods.

In Chapter 3, we propose an efficient machine learning algorithm called NadaBoost, which is applicable to visual learning frameworks because of its capacity to improve boosting algorithm. Since the problem of AdaBoost stems from the fact that its learning strategy heavily weights hard examples in order to classify them correctly, we determine weight threshold by searching for the optimal threshold using an efficient search algorithm and reduce the weights of those hard examples which have higher weights than this optimal threshold. We verify the effectiveness of the proposed method through several data classification experiments using various benchmark data sets.

In Chapter 4, we propose a visual learning method for object recognition tasks. To discriminate objects, we utilize the intensity values of the pixels in an image as features. We then construct hypotheses to classify given images using the Set Covering Machine algorithm. Since an image contains a vast number of pixels, using all pixels makes the learning and recognition processes quite time-consuming. Thus, for efficient learning and recognition, we introduce feature selection. Specifically, we define an estimation criterion to assess the usefulness of each pixel and select a small number of useful pixels which contribute to discriminate objects correctly according to the estimation criterion. As a result of this feature selection, the computational complexity is considerably decreased without the degradation of the recognition performance. Through several recognition experiments, we verify the performance of the proposed method.

In Chapter 5, we propose an efficient visual learning model which is able to utilize diverse features such as contours, textures and spatial frequency to discriminate complex objects accurately. In order to deal with multiple features, we introduce ensemble learning. We construct multiple classifiers using diverse features and integrate them into an ensemble classifier which performs better than a single classifier. Ensemble learning generally trains each classifier separately and integrates only the learning results of multiple classifiers. This learning strategy is insufficient because there is no interaction among multiple classifiers. In the visual learning process, all classifiers should help each other to optimize their own individual recognition performance. Thus, we introduce this cooperative learning strategy through the interaction among classifiers. This strategy results in a better ensemble classifier compared with existing ensemble learning methods. Since we attempt the correct classification of real-world images which contain hard examples such as deformed or occluded objects, it is necessary to deal with such hard examples. Thus, we introduce NadaBoost, proposed in Chapter 3, to detect hard examples and prevent them from affecting recognition performance. We then demonstrate the effectiveness of this strategy through several object recognition experiments and a comparison with several existing object recognition methods.

Finally, Chapter 6 remarks on the effectiveness and remaining problems of the proposed methods and refers to future work.

Chapter 2

Visual Learning

At the beginning of computer vision and image understanding research, the recognition process was human-intensive. That is, large part of domain knowledge is given by hand. As the amount and complexity of images increase, however, conventional image recognition methods have difficulty in recognizing real-world objects because of the following problems: First, it is quite difficult to give domain-specific knowledge manually for complex recognition problems. For example, predefined templates are needed by template matching methods. However, real-world objects have a huge variety of shapes and textures and manually defining appropriate templates is virtually impossible. Second, although the recognition performance of the conventional methods is sufficient for only limited domains (e.g., face recognition and hand-written character recognition), they have great difficulty in effecting flexible recognition, which is able to discriminate any real-world object. Finally, the recognition performance of these methods tend to be affected by noisy images which contain cluttered backgrounds, bad lighting conditions, occluded or deformed objects, and so on.

To solve these problems and improve recognition ability, visual learning has been developed. The first problem can be solved based on the property of machine learning that the resulting learning model consists of domain-specific knowledge derived through the analysis of given images called training examples. Although the features in the data used for machine learning is explicitly given in advance, the features of image data are generally not specified because an image is given just as a set of pixels. Thus, some kinds of features are generated by visual learning algorithms using feature extraction and/or feature construction techniques. Since various types of features are generated from image data, flexible recognition can be performed by generating and utilizing appropriate features according to the given recognition tasks. Consequently, the second problem can be solved. As for the third problem, the statistical distributions of the features generated from noisy images, which contain occlusion, deformation, bad lighting conditions, tend to be different from the images which contain none of them. Thus, in the statistical analysis, they can be detected as outliers. Machine learning is able to prevent the degradation of recognition performance by eliminating these outliers.

Since various methods of computer vision processing and machine learning have been proposed, diverse visual learning models can be developed by combining these methods. Therefore, visual learning is actively studied and many visual learning models have been proposed. The models of visual learning can be defined from the following viewpoints:

1. Representation of examples
2. Features
3. Classifiers

The first is the method of representing an example (i.e., an image). Since an image contains a large amount of information, it is represented by a data structure. In visual learning, an image is often represented by one or more feature vectors because this representation is suitable to enable most machine learning algorithms to handle and analyze images. The second is the features used to discriminate objects. A variety of features can be extracted from an image using feature extraction methods such as edge detector and local feature descriptor [18]. In general, multiple features are used to describe complex objects, especially in visual learning methods based on ensemble learning. The third is the classifiers which detect the objects in images. The number of classifiers and the methods of training each classifier are closely related to the recognition performance of visual learning methods. According to this definition, we divide visual learning models into four types as shown in Figure 2.1.

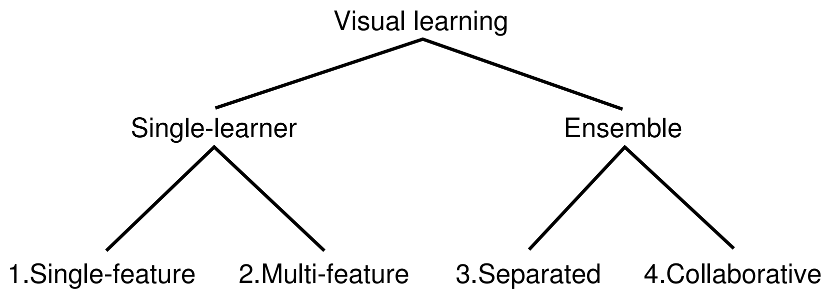


Figure 2.1: Visual learning models.

The first model is the *single-feature model* [37]. This is the simplest visual learning model as it discriminates objects using a single feature and a single classifier. This model is defined as follows:

1. An example is represented by *a single* feature vector.
2. *A single* feature is used.

3. *A single* classifier is trained.

For example, in [37] faces are described by a feature vector in a multi-dimensional space called face space, which is defined using only the intensity values of the pixels in the face image. Although this method achieves high recognition accuracy, it lacks flexibility and is insufficient for generic object recognition because a single feature is inadequate when describing a wide variety of objects.

The second is the *multi-feature model* [15]. This model has a greater ability to describe complex objects because of its utilization of multiple features. Thus, it can perform more flexible recognition than the single-feature model. This model is defined as follows:

1. An example is represented by *multiple* feature vectors.
2. *Multiple* features are used.
3. *A single* classifier is trained.

For example, in [15], a classifier is trained using multiple features obtained using image filtering and a logical/mathematical calculation of intensity values. The optimal combination of features is automatically determined by genetic programming.

Although the recognition performance is to some extent improved by multiple features, the representational capability of a single classifier is inadequate for dealing with various features. Hence, the *separated ensemble model* [38] [24] is developed. In this model, a single classifier utilizes one or several features. Based on ensemble learning, multiple classifiers are constructed. By combining multiple classifiers which utilize different features, this model takes advantage of multiple features. This model is defined as follows:

1. An example is represented by *multiple* feature vectors.
2. *A single or multiple* feature(s) are used.
3. *Multiple* classifiers are *separately* trained.

For example, in [38], a successful object detection method is proposed. In this method, a cascade classifier, which is an ensemble of classifiers, is constructed based on a single feature. The cascade classifier is able to select a small number of useful features from a large number of extracted features, and thus can quickly and accurately detect objects. In [24], appearance-based and region-based classifiers are trained using decision trees and discriminant analysis. The two classifiers are then integrated into a classifier that achieves much better recognition performance than a single classifier.

Because conventional ensemble learning methods do not take hard examples into consideration, however, the recognition performance of separated ensemble models can be degraded by the influence of hard examples. Thus, overfitting is a crucial problem in the presence of many hard examples. In order to solve this problem, the ensemble model defined as follows is required:

1. An example is represented by *multiple* feature vectors.
2. *Multiple* features are used.
3. *Multiple* classifiers are *interactively* trained.

Based on this definition, we propose a *collaborative ensemble model.* In the collaborative ensemble model, multiple classifiers are simultaneously trained. Thus, when a classifier X has misclassified some examples, these examples are subsequently learned by other classifiers instead of by X. Conversely, X learns examples that have been misclassified by other classifiers. This strategy facilitates finding and thus dealing with hard examples. In addition, each classifier is able to optimize its own performance through the collaboration process. As a result, the collaborative ensemble model performs more efficient learning than do separated ensemble models.

Chapter 3

Noise-robust Boosting Algorithm: NadaBoost

3.1 Introduction

Boosting is a method for improving the classification accuracy of a given learning algorithm by combining hypotheses created by the learning algorithm. Recently, many researchers have carried out research on AdaBoost [11][31][33], the typical boosting algorithm. They have reported that AdaBoost has shown very good performance on many classification problems.

However, some drawbacks of AdaBoost have also been reported. One of the drawbacks of AdaBoost is its susceptibility to noisy examples. In a phenomenon called overfitting, AdaBoost tends to concentrate on noisy examples, thus worsening its performance. AdaBoost usually calls repeatedly for a given learning algorithm, which we term WeakLearn. AdaBoost maintains a set of weights over the training examples which is employed as a distribution. Initially, all weights are set equally in advance, but on each round, the weights of incorrectly classified examples are increased so that WeakLearn is forced to concentrate on those examples which are difficult to classify correctly. Thus, as the weight of a certain example becomes higher, the hypothesis becomes more specific to the example. However, since noisy examples are quite difficult to classify correctly, AdaBoost tends to assign higher weights to these examples if any are included in the training set. As a result, AdaBoost overfits the noise [7].

To solve this problem, several boosting methods have been proposed to improve the algorithm of AdaBoost. For example, MadaBoost [9] reduces the influence of noisy examples by modifying AdaBoost's weighting rule of AdaBoost so that it does not give high weights to noisy examples. From the viewpoint that the learning process of AdaBoost can be regarded as a margin maximization process, Soft Margin AdaBoost has been proposed [30]. It has been proven in [30] that AdaBoost increases the margins not only of non-noisy examples but also of noisy examples, with the result that overfitting occurs. Soft Margin AdaBoost prevents overfitting by controlling the margins of noisy examples. Since these methods implicitly control the weights or margins of noisy examples, they are unable to detect these examples. In order to reduce fully the influence of overfitting, these methods selectively decrease the weights of noisy examples, which is inadequate to reduce their influence.

To resolve this issue, we propose an effective modification of the weighting rule of AdaBoost which overcome its vulnerability to noisy examples. Since AdaBoost tends to assign higher weights to noisy examples, those examples to which excessively high weights are assigned seem to be noisy examples. Therefore, we assume that, when too highly weighted examples are misclassified, the weights of such examples should be decreased. Thus, we introduce thresholds to judge whether or not the weight is too high. We also modify the weighting rule so that weights that are higher than the threshold are always decreased even if the examples are misclassified. This modification prevents WeakLearn from overfitting noisy examples.

However, it is very difficult to choose the optimal threshold because quite large time complexity is required to search the optimal thresholds from all possible thresholds. Thus, we apply the beam search, which is a variant of the best first search, to our method. In the search process, the search space is appropriately limited by the beam search. This search strategy leads to the determination of weight threshold in realistic time complexity without the degradation of classification accuracy.

3.2 AdaBoost

AdaBoost, which was first proposed by Freund and Schapire [11], is the typical boosting algorithm. The generalized version of AdaBoost was introduced by Schapire and Singer [33]. The pseudocode of Schapire and Singer's AdaBoost is shown in figure 3.1.

The pseudocode takes a training set of m examples $(x_1, y_1), \cdots, (x_m, y_m)$ as input where each $x_i (1 \leq i \leq m)$ is an element of *instance space* X and each y_i is an element of *label space* Y. To make it simple, we assume $Y = \{-1, +1\}$; that is, the binary classification problem. AdaBoost repeatedly calls WeakLearn. The number of iterations T is determined in advance. AdaBoost maintains a set of weights over training examples which is employed as a distribution. The weight of this distribution on training example i on the t-th round is denoted as $D_t(i)$. On each round, the distribution D_t is normalized by the normalization factor Z_t, so $D_t(i) \in [0, 1]$. On the tth round, WeakLearn creates a hypothesis $h_t : X \rightarrow \mathbb{R}$ based on the distribution. We call the hypothesis a weak hypothesis. In this research, we limit $h_t : X \rightarrow [-1, +1]$. Weights, which are initially equal, are updated on each round. The weights of correctly classified examples are decreased and those of misclassified examples are increased. Once the weak hypothesis h_t has been received, AdaBoost chooses a parameter α_t, which intuitively measures the importance of h_t. α_t is employed when the weights are updated. In Schapire and Singer's AdaBoost, α_t is given as follows:

```
Given: $(x_1, y_1), \cdots, (x_m, y_m);\ x_i \in X, y_i \in \{-1, +1\}$.
Initialize $D_1(i) = 1/m$.
For $t = 1, \cdots, T$:

    1. Train WeakLearn using the weights $D_t$.
    2. Get weak hypothesis $h_t : X \to [-1, +1]$.
    3. Choose $\alpha_t \in \mathbb{R}$.
    4. Update:

        $D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$,

    where $Z_t$ is a normalization factor
    (chosen so that $D_{t+1}$ will be a distribution).

Output the final hypothesis:

    $H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$.
```

Figure 3.1: Schapire and Singer's AdaBoost.

$$\alpha_t = \frac{1}{2} \ln \left(\frac{1 + r_t}{1 - r_t} \right), \tag{3.1}$$

where

$$r_t = \sum_i D_t(i) y_i h_t(x_i). \tag{3.2}$$

After the iteration is finished, AdaBoost combines all weak hypotheses as a *final hypothesis* H. H is the weighted majority vote of weak hypotheses where α_t is the weight assigned to h_t. H is given as follows:

$$H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x) \right). \tag{3.3}$$

This weighting rule described above is the advantage of AdaBoost; however it can also be considered the drawback. The training set usually includes noisy or exceptional examples, called hard examples, with which it is undesirable for learning algorithms to be specialized. As Dietterich has reported [7], AdaBoost tends to overfit hard examples by weighting them much more heavily than other examples. As a result, weak hypotheses are specialized with hard examples, causing the final hypothesis to overfit hard examples. In order to prevent AdaBoost from overfitting, we have to restrict the increment of the weights assigned to hard examples. That is, if the training set includes hard examples, we have to detect hard examples among the misclassified examples and modify the weighting rule in order to decrease weights of misclassified hard examples.

3.3 NadaBoost

3.3.1 Modification of the weighting rule

We employ each weight as the indicator in order to judge whether or not the example is a hard example. We assume that examples which have already been given too high weights are treated as hard examples. Since it is preferable that AdaBoost should not assign much higher weights to such examples, we introduce thresholds to judge whether or not weights are too high. Thus, we modify the weighting rule as follows:

$$\begin{aligned} D_{t+1}(i) &= \frac{D_t(i)}{Z_t} \cdot \begin{cases} e^{-\alpha_t y_i h_t(x_i)}, & h_t(x_i) y_i \geq 0, \\ e^{-\alpha_t y_i h_t(x_i)}, & h_t(x_i) y_i < 0 \wedge D_t(i) \leq HighWeight_t, \\ e^{\alpha_t y_i h_t(x_i)}, & h_t(x_i) y_i < 0 \wedge D_t(i) > HighWeight_t \end{cases} \\ &= \frac{D_t(i)\exp(-\alpha_t y_i h_t(x_i) I(y_i h_t(x_i), D_t(i)))}{Z_t}, \qquad (3.4) \end{aligned}$$

where

$$I(y_i h_t(x_i), D_t(i)) = \begin{cases} -1, & h_t(x_i) y_i < 0 \wedge D_t(i) > HighWeight_t, \\ +1, & \text{otherwise.} \end{cases} \qquad (3.5)$$

Now, let $HighWeight_t$ be the threshold on the tth round. If a certain training example is classified correctly, the weight is decreased as under the weighting rule of AdaBoost. If the example is misclassified, the weight is compared with $HighWeight_t$. The weight is increased only when the weight is lower than $HighWeight_t$. If the weight is higher than $HighWeight_t$, the weight is decreased in the same way as if the training example were classified correctly. Thus, even if hard examples are misclassified on every round, their weights are not excessively high. We now analyze the performance of our modification. Our analysis is closely modeled on that of Schapire and Singer [33]. We give and prove the upper bound of the training error of the modified version of AdaBoost as theorem 1.

Theorem 1 *The training error of the modified weighting rule is at most*

$$\prod_{t=1}^{T} Z_t \cdot \left\{ \sum_{i=1}^{m} D_{T+1}(i) \cdot \exp\left(-2 \sum_{t: I_{t,i}=-1} \alpha_t y_i h_t(x_i) \right) \right\}. \qquad (3.6)$$

Proof. To simplify the notation, let $I_{t,i}$ be $I(y_i h_t(x_i))$. From equation (3.4), we obtain

$$D_{T+1}(i) = \frac{\exp(-\sum_t \alpha_t y_i h_t(x_i) I_{t,i})}{m \prod_t Z_t}. \tag{3.7}$$

Here, if $I_{t,i} = -1$ then $y_i h_t(x_i) < 0$. Thus,

$$D_{T+1}(i) = \frac{\exp(-\sum_t \alpha_t y_i h_t(x_i) + 2\sum_{t:I_{t,i}=-1} \alpha_t y_i h_t(x_i))}{m \prod_t Z_t}. \tag{3.8}$$

Let $f(x) = \sum_t \alpha_t h_t(x)$. Then we obtain

$$D_{T+1}(i) = \frac{\exp(-y_i f(x_i)) \cdot \exp(2\sum_{t:I_{t,i}=-1} \alpha_t y_i h_t(x_i))}{m \prod_t Z_t}. \tag{3.9}$$

Moreover, let $[\pi]$ be an indicator variable. $[\pi]$ is 1 if the predicate π is true and 0, otherwise. Then we obtain the following:

$$\begin{array}{rcl}
H(x_i) \neq y_i & \Rightarrow & y_i f(x_i) \leq 0 \\
& \Rightarrow & \exp(-y_i f(x_i)) \geq 1 \\
& \Rightarrow & [H(x_i) \neq y_i] \leq \exp(-y_i f(x_i)), \quad \text{and} \\
H(x_i) = y_i & \Rightarrow & y_i f(x_i) \geq 0 \\
& \Rightarrow & \exp(-y_i f(x_i)) \geq 0 \\
& \Rightarrow & [H(x_i) \neq y_i] \leq \exp(-y_i f(x_i)).
\end{array} \tag{3.10}$$

Thus, in all cases we obtain

$$[H(x_i) \neq y_i] \leq \exp(-y_i f(x_i)). \tag{3.11}$$

Here,

$$error(H(x)) = \frac{1}{m} \sum_{i=1}^{m} [H(x_i) \neq y_i]. \tag{3.12}$$

Combining equations (3.9) and (3.11), we obtain the following:

$$\begin{aligned}
\frac{1}{m} \sum_{i=1}^{m} [H(x_i) \neq y_i] &\leq \frac{1}{m} \sum_{i=1}^{m} \exp(-y_i f(x_i)) \\
&= \frac{1}{m} \sum_{i=1}^{m} D_{T+1}(i) \cdot m \prod_t Z_t \cdot \exp\left(-2 \sum_{t:I_{t,i}=-1} \alpha_t y_i h_t(x_i)\right) \\
&= \prod_t Z_t \cdot \left\{ \sum_{i=1}^{m} D_{T+1}(i) \cdot \exp\left(-2 \sum_{t:I_{t,i}=-1} \alpha_t y_i h_t(x_i)\right) \right\}.
\end{aligned} \tag{3.13}$$

In equation (3.13), we cannot determine the best value of the parameter α_t directly. On each round, we should choose α_t , which minimizes the upper bound. However, the second term of the upper bound $\sum_{i=1}^{m} D_{T+1}(i) \cdot \exp(-2\sum_{t:I_{t,i}=-1} \alpha_t y_i h_t(x_i))$ is not determined unless the iteration is finished because the term $\sum_{t:I_{t,i}=-1} \alpha_t y_i h_t(x_i)$ is the sum of $\alpha_t y_i h_t(x_i)$ on each round. Thus, it is impossible to choose α_t for the purpose of minimizing the upper bound. Therefore, we choose α_t, so that the first term of the right-hand side of equation (3.13), $\prod_t Z_t$, can be optimized.

3.3.2 Choosing α_t

In order to optimize $\prod_t Z_t$, we choose α_t, which minimizes Z_t on each round. On the tth round, we have

$$Z_t = \sum_{i=1}^{m} D_t(i) \exp(-\alpha_t y_i h_t(x_i) I_{t,i}). \tag{3.14}$$

By following Schapire and Singer [33], we omit the subscript t from each notation and let $u_i = y_i h_t(x_i)$ and $I_i = I(y_i h(x_i), D(i))$. Here, $u_i \in [-1, +1]$, since weak hypotheses $h \in [-1, +1]$. Then, we have

$$\begin{aligned} Z &= \sum_i D(i) \exp(-\alpha u_i I_i) \\ &= \sum_{i:I_i=1} D(i) e^{-\alpha u_i} + \sum_{i:I_i=-1} D(i) e^{\alpha u_i} \\ &\le \sum_{i:I_i=1} D(i) \left(\frac{1+u_i}{2} e^{-\alpha} + \frac{1-u_i}{2} e^{\alpha} \right) \\ &\quad + \sum_{i:I_i=-1} D(i) \left(\frac{1-u_i}{2} e^{-\alpha} + \frac{1+u_i}{2} e^{\alpha} \right). \end{aligned} \tag{3.15}$$

By solving the equation $\partial Z/\partial\alpha = 0$, we find that Z is minimized when

$$\alpha = \frac{1}{2} \ln \left(\frac{1+r'}{1-r'} \right), \tag{3.16}$$

where

$$r' = \sum_i D(i) u_i I_i. \tag{3.17}$$

Substituting this optimal α in equation (3.15), we have

$$Z \le \sqrt{1 - r'^2}. \tag{3.18}$$

Thus, the upper bound of training error of our method is the following.

Theorem 2 *Assume that each h_t has range $[-1,+1]$ and we choose*

$$\alpha_t = \frac{1}{2}\ln\left(\frac{1+r'_t}{1-r'_t}\right), \tag{3.19}$$

where

$$r'_t = \sum_i D_t(i) y_i h_t(x_i) I_{t,i}. \tag{3.20}$$

Then the training error of our method is at most

$$\begin{aligned} error(H(x)) \quad \leq \quad & \left(\prod_{t=1}^{T}\sqrt{1-r'^2_t}\right) \\ & \cdot\left(\sum_{i=1}^{m} D_{T+1}(i)\cdot\exp\left(-2\sum_{t:I_{t,i}=-1}\alpha_t y_i h_t(x_i)\right)\right). \end{aligned} \tag{3.21}$$

We call this upper bound *errorbound*. As the threshold $HighWeight_t$ becomes lower, the first term of the righthand side of the equation (3.21) becomes smaller and the second term becomes larger. Intuitively, we can say that the equation (3.21) represents the improvement of classification accuracy by avoiding the concentration on excessively hard examples and the degradation of classification accuracy for such examples, respectively. However, the increment of the second term is exponential. Thus, if $HighWeight_t$ is too low, the increment of the second term offsets the decrement of the first term. In other words, too low $HighWeight_t$ worsens the performance of our method. Thus, we have to consider the value of errorbound when we set $HighWeight_t$ on each round.

3.3.3 Setting HighWeight

In this research, we employ the upper bound of training error as the indicator of the performance of learning algorithms. The upper bound of AdaBoost was given by Schapire and Singer [33]. We show the upper bound as theorem 3.

Theorem 3 *(Schapire and Singer). Assume that each h_t has a range $[-1,+1]$ and that we choose*

$$\alpha_t = \frac{1}{2}\ln\left(\frac{1+r_t}{1-r_t}\right), \tag{3.22}$$

where

$$r_t = \sum_i D_t(i) y_i h_t(x_i). \tag{3.23}$$

Then the training error of H is at most

$$\prod_{t=1}^{T} \sqrt{1 - r_t^2}. \tag{3.24}$$

We call this upper bound *errorbound_ada*. On each round, we set $HighWeight_t$ so that the upper bound of our method must be superior to that of AdaBoost. That is, *errorbound* satisfies the following inequality:

$$errorbound \leq errorbound\_ada. \tag{3.25}$$

By setting each $HighWeight_t$ in the above way, our method can generate theoretical evidence that shows the superiority to AdaBoost.

The value of $HighWeight$ is continuous since the weights of training examples are also continuous. However, we can limit the range of $HighWeight$. For example, if $D_t(i) \in \{0.1, 0.2, 0.3\}$, the possible thresholds are any one of "(1) All training examples are regarded as hard examples", "(2) Training examples with the weights 0.2 or 0.3 are regarded as hard examples", "(3) Training examples with the weights 0.3 are regarded as hard examples", or "(4) No training examples are regarded as hard examples." We show the examples of the possible thresholds in figure 3.2.

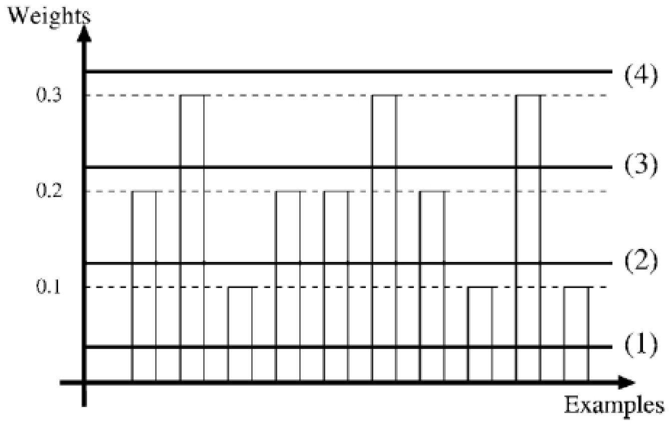


Figure 3.2: The examples of possible weight thresholds.

However, we can neglect the first threshold because it is impossible that all training examples would be hard examples. Thus, in the above example, we can consider three possible thresholds, which we call *threshold candidates.*

Specifically, we use the possible value of the weights as the candidates of thresholds. For example, if $D_t(i) \in \{0.1, 0.2, 0.3\}$, the candidates of thresholds are 0.1, 0.2, and 0.3. When 0.1 is chosen as $HighWeight$, all weights higher than 0.1 are decreased.

Here, the weights influence the weak hypotheses, the parameter α, and so on. Furthermore, the weighting rule is dependent on $HighWeight$. Thus, the state of each round depends on the value of $HighWeight$. In the above example, then, when the number of the candidates of thresholds is three, the number of possible states on the next round is also three. Thus, by regarding the states as nodes and the candidates of thresholds as branches, we can describe the state transition as a tree structure. We show an example of the tree structure in figure 3.3. The three branches of $S_{t-1}[j]$ correspond to the examples of the candidates of thresholds described above. We search the final state that satisfies the inequality (3.25) and let the path to the final state be $HighWeight$ on each round.

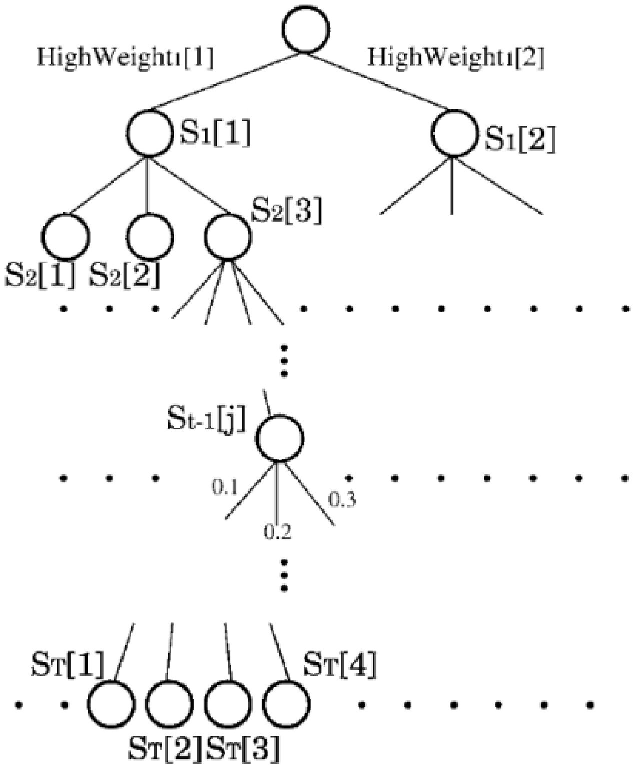


Figure 3.3: The tree structure of the beam search.

However, searching all of the tree structure requires quite large time complexity. In order to solve this problem, we use the beam search, a variant of the best first search. The beam search also opens the node with the lowest cost value in the best first search manner. The difference lies in the fact that the number of nodes that the beam search can retain is limited, whereas the number of nodes that the best first search can retain is unlimited. The bound of retained nodes is called *beam_width*. If the number of nodes that are to be retained is higher than the value of *beam_width*, then the beam search gives preference to those nodes that have a lower cost.

As a cost function of the beam search, we employ the estimate value of the upper bound of training error. The estimate value on the t-th round is represented as $estimated\_error_t$. We define $estimated\_error_t$ as follows:

$$\begin{aligned} estimated\_error_t &= \left(\prod_{k=1}^{t} \sqrt{1 - r_k'^2} \right) \\ &\quad \cdot \left\{ \sum_{i=1}^{m} D_{t+1}(i) \exp \left(-2 \sum_{k:I_{k,i}=-1} \alpha_k y_i h_k(x_i) \right) \right\}. \end{aligned} \tag{3.26}$$

Equation (3.26) means the upper bound of training error, *errorbound*, on the tth round. If a certain node has a higher *estimated_error*, then the child node also seems to have higher *estimated_error*. Thus, the final state also seems to have higher errorbound. By applying the beam search, those nodes that have high *estimated_error* are not opened.

3.3.4 Entire Algorithm

We propose the new boosting algorithm NadaBoost. NadaBoost updates the weights of training examples by the modified weighting rule described above. We show the entire algorithm of NadaBoost in figure 3.4.

At step 1, WeakLearn creates a weak hypothesis h_t. At step 2, let $HighWeight_t[j]$ be the jth threshold on the tth round. We must determine each of the possible values of the threshold. At step 3, α and D_{t+1} are determined by considering each $HighWeight_t[j]$ and are given from the modified weighting rule (1) and theorem 2. We calculate the value of the $estimated\_error_t$ by equation (3.26). Let the state be determined by employing $HighWeight_t[j]$ be $S_t[j]$. The state means the values of the all variables that appear in NadaBoost.

At step 4, all of $S_t[j]$ are added to *S_list*. However, the number of retained states is limited by the value of *beam_width*. For example, we assume that NadaBoost tries to add three states to *S_list* on the condition that the value of *beam_width* equals 2. In this case, the state which has the highest *estimated_error* is erased. At step 5, $S_k[j]$ that has the lowest $estimated\_error_k$ is popped out from *S_list*. If $S_k[j]$ is not the state of the final round, i.e., if $k \neq T$, NadaBoost computes the next state $S_{k+1}[j']$, where j' identifies the possible value of the threshold. If $S_k[j]$ is the state of the final round, that is, $k = T$, we compare errorbound of $S_k[j]$ with *errorbound_ada*. We find $S_k[j]$ that satisfies the following:

$$errorbound \leq errorbound\_ada \tag{3.27}$$

```
Given: $(x_1, y_1), \cdots, (x_m, y_m)$;  $x_i \in X, y_i \in \{-1, +1\}$.
Initialize $D_1(i) = 1/m, t = 0, end = 0$.
Let the number of iterations be $T$.
Let the number of states which NadaBoost can retain be $beam\_width$.

While $end \neq 1$:
    1. Train WeakLearn using the weights $D_t$
       and get weak hypothesis $h_t : X \to [-1, +1]$.
    2. Determine candidates of the threshold $HighWeight_t[j]$.
    3. For all $HighWeight_t[j]$:
       (1) Determine $\alpha_t$ and $D_{t+1}$;
       (2) Determine $estimated\_error_t$;
       (3) Let the state be $S_t[j]$.
    4. Add all $S_t[j]$ to $S\_list$.
    5. If $S\_list$ is empty, then let $end$ be 1.
       Otherwise, then choose $S_k[j]$ which has the lowest $estimated\_error$ and:
       (1) Pop out $S_k[j]$ from $S\_list$.
       (2) If $k = T$, then
            If $errorbound \leq errorbound\_ada$,
            Then add $S_k[j]$ to $Goal\_list$ and go to step 5.
            Otherwise, then let $t$ be $k + 1$.

Choose $S_k[j]$ which has the lowest $errorbound$ from $Goal\_list$.
Output the final hypothesis:

    $H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$.
```

Figure 3.4: NadaBoost algorithm.

and add such $S_k[j]$ to *Goal_list*. That is, the states with *errorbound* superior to AdaBoost are collected in *Goal_list*. NadaBoost repeats a series of steps followed until *S_list* becomes empty. Finally, NadaBoost chooses the state that has the lowest *errorbound* from *Goal_list* and creates a final hypothesis based on the state.

3.4 Experiments

In this section, we show the results of running AdaBoost and NadaBoost. We employed ten data sets drawn from the UCI Machine Learning Repository and the **maze** data set which we prepared. We employed C4.5 [29] as WeakLearn and performed a ten-fold cross-validation paired t test [6] in order to evaluate each of the two methods. We set the number of iterations and *beam_width* for NadaBoost at 30. For comparison, we also ran C4.5 alone. The results of the experiment are shown in table 3.1. The mark * in the t test column represents the fact that NadaBoost outperforms AdaBoost significantly.

Table 3.1: The results of experiments for UCI data sets.

Data set	Error(%)			
name	NadaBoost	AdaBoost	C4.5	t test
breast-w	3.72	4.15	6.16	*
crx	13.77	13.91	17.25	-
german	25.80	25.90	26.60	-
heart-c	21.81	22.80	28.05	*
hepatitis	18.08	20.71	21.21	*
ionosphere	5.42	6.55	9.11	*
maze	5.83	6.48	18.39	-
sonar	19.17	22.05	35.52	*
tic-tac-toe	1.56	1.67	15.13	-
pima-indians	26.41	27.06	26.81	*
vote	4.83	5.29	4.83	-

In six data sets, it is judged that NadaBoost outperformed AdaBoost significantly. In the other data sets, the performance of NadaBoost was superior or equal to that of AdaBoost. Thus, it seems that NadaBoost is superior to AdaBoost.

The data set which shows the tendency of overfitting is pima-indians. On pima-indians, the performance of AdaBoost is inferior to that of C4.5. However, the performance of NadaBoost is superior to that of C4.5, causing the modified weighting rule to prevent the overfitting of hard examples. In addition, we added 10% noise to the tic-tac-toe and maze data sets, which included no noisy examples, and we performed the same experiments as above. The results of these experiments are shown in table 3.2.

Table 3.2: The results of experiments for noisy data sets.

Data set	Error(%)			
name	NadaBoost	AdaBoost	C4.5	t test
maze	24.46	25.76	26.61	*
tic-tac-toe	21.30	23.37	27.56	*

Because of the added noisy examples, the performance of AdaBoost worsens. While NadaBoost's performance also worsens, however, its degeneration in performance is smaller than that of AdaBoost. We thus find that the modified weighting rule prevents the overfitting of noisy examples as shown in table 3.1.

3.5 Related works and comparison with them

In this section, we mention other boosting algorithms based on the margins [32] of the training examples. We also analyze the margins in our algorithms, and compare the result with other algorithms. The margin is the value which shows the confidence in the prediction for a certain training example. The margin of the example (x, y) is defined as follows:

$$\frac{y\sum_{t=1}^{T}\alpha_t h_t(x)}{\sum_{t=1}^{T}\alpha_t}. \tag{3.28}$$

Obviously, the margin is a real number in $[-1, +1]$ and it is positive if the final hypothesis $H(x)$ correctly classifies the example. The example whose margin is close to $+1$ is classified correctly by many weak hypotheses; conversely, the example whose margin is close to -1 is misclassified by many weak hypotheses. That is, the magnitude of the margin can be regarded as a criterion of confidence in the prediction of the final hypothesis.

Schapire et al. [32] proved that larger margins on the training examples can be translated into a superior upper bound of the generalization error. They also showed that AdaBoost was effective in maximizing the margins of training examples. A few rounds later, AdaBoost makes all the margins positive: that is, the final hypothesis $H(x)$ correctly classifies all the training examples. If the training set includes noisy examples, it seems that this characteristic of AdaBoost causes overfitting. Some research based on the notion described above has been conducted.

Mason et al. [23] showed that some voting method such as AdaBoost minimize the cost function by the gradient descent method. They also proposed the DOOM 2 algorithm, which performs a gradient descent optimization of the cost function of the margin. They have argued that AdaBoost is a special case of their algorithm. Rätsch et al. [30] showed that AdaBoost is a process which minimizes a cost function $g(b)$ and proved that the weights on the training set are given by the margins and $g(b)$. They showed that AdaBoost increases the margins of all training examples excessively, thus overfitting hard examples, and proposed a new boosting algorithm to decrease the lower bound of the margins.

As we described above, AdaBoost makes all the margins positive. In contrast, the margins of some training examples are negative on the other algorithms. This phenomenon shows that these algorithms eliminate the influence of hard examples based on the final hypothesis by giving up their attempts to classify these examples.

For comparison with these algorithms, we analyzed the behavior of NadaBoost in terms of the margins for noisy environments. We implemented AdaBoost and NadaBoost for the tic-tac-toe and maze data set with 10% noise. We set the number of iteration and *beam_width* for NadaBoost at 30. Figure 3.5 shows the margin distribution for this experiment.

Figure 3.5 does not reveal an obvious difference between AdaBoost and NadaBoost. In contrast to the two algorithms described above, most of the margins

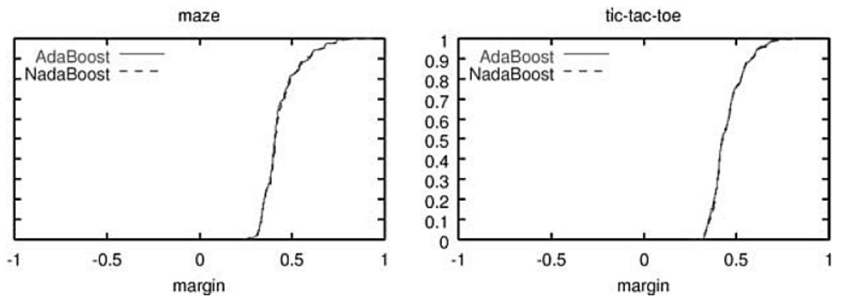


Figure 3.5: The margin distributions for AdaBoost and NadaBoost.

Table 3.3: The mean value of margins.

Data set name		Mean value	
		Normal	Noise
maze	AdaBoost	0.440651	0.364279
	NadaBoost	0.444992	0.351959
tic-tac-toe	AdaBoost	0.457592	0.387820
	NadaBoost	0.461568	0.373711

are still positive. Table 3.3 shows the mean value of the margins.

From table 3.3, we can find that NadaBoost effectively decreases the margins of noisy examples. As we described above, the magnitude of the margin shows the confidence of the prediction. From these results, we can find that NadaBoost does not give up trying to classify hard examples but instead weakens their influence. Therefore, NadaBoost can avoid overfitting without making some margins negative.

For the experiments in this section, we used the training examples with 10% noise artificially added. However, we cannot generally recognize whether or not a data set includes noise. We can only recognize that some examples may be noisy. We can conclude that NadaBoost classifies examples along such a point of view. That is, NadaBoost classifies examples without determining that some examples assigned to too high weights are noisy.

3.6 Conclusion

We have described the modification of AdaBoost and proposed NadaBoost. Then, we formulated NadaBoost's upper bound of training error. We have provided empirical evidence that NadaBoost has lower error than AdaBoost through several experiments. Thus, we can claim that NadaBoost is effective in preventing the overfitting of the noisy data. Since the computational complexity of NadaBoost is much higher than AdaBoost because of its search process, we should try to devise a more efficient search strategy to determine the optimal (or suboptimal) value of $HighWeight$.

Chapter 4

Visual Learning based on Set Covering Machine

4.1 Introduction

In this chapter, we propose an object recognition method which is able to flexibly and accurately distinguish real-world objects based on visual learning. As mentioned in Chapter 2, visual learning [15] is widely used to acquire information and knowledge from image data for various object recognition tasks.

There are, however, some open problems for visual learning. Firstly, many conventional visual learning methods are based on trial-and-error learning algorithms, such as genetic programming [14] and reinforcement learning [26]. These methods require considerable learning time to search for the optimal solution. In addition, their learning process is nondeterministic and depends to some extent on randomness. Thus, their learning results can be rather unstable. Secondly, because an image consists of a large number of pixels, it contains a large amount of information. Although it is effective to utilize the intensity distribution of an image as its feature, the computational cost of analyzing each pixel is a crucial problem for efficient recognition even if the size of the image is not very large. Finally, given images often contain noise, a factor which affects the recognition performance when using the intensity distributions of pixels. To reduce the influence of noise, image filtering is effective. However, the number of applicable filters is large, and it is difficult to determine the appropriate combination of filters.

To solve these problems, we introduce the Set Covering Machine (SCM) [19][20] into visual learning. The SCM discriminates objects using the distribution statistics of pixel intensity values. Since the SCM works without trial and error, the cost of searching for the optimal solution is relatively small, while the result of learning is stable. Additionally, the SCM is a general-purpose learning algorithm and is thus applicable to a variety of visual learning tasks without domain-specific knowledge.

Since utilizing all pixels is quite time-consuming and many redundant pixels are included in each image, we introduce a feature selection method to reduce the number of attributes (i.e., pixels) used by the SCM. We define an estimation function to evaluate the usefulness of an attribute and use this function to select a small number of useful attributes. This strategy can reduce the computational complexity considerably. In addition, we propose the adaptive learning method based on the MDL principle to make the hypothesis more concise.

We perform image filtering for each given image using several filters to reduce the influence of noise. Since the combination of filters is exponential, we efficiently determine the appropriate filters using the beam search. The beam search can restrict the search space and improve the efficiency of finding a combination of filters without the degradation of recognition performance. Through the experiment using real-world image data, we verify the effectiveness and efficiency of the proposed method.

4.2 Set Covering Machine

The SCM [19][20] is a general-purpose learning algorithm that is applicable to various learning tasks. The SCM takes a set of m examples $\{(\mathbf{x}_1, y_1),\ (\mathbf{x}_2, y_2),\ \cdots,\ (\mathbf{x}_m, y_m)\}$ as the training set, where each $\mathbf{x}_i$ is a n-dimensional vector and each y_i is a class label, which is an element of the label space $Y = \{0, 1\}$.

In the SCM, a Boolean-valued function h_i is defined for each example $\mathbf{x}_i$. h_i is called *feature*. A feature is a classifier to be defined as:

$$h_{i,\rho}(\mathbf{x}) \overset{\text{def}}{=} h_\rho(\mathbf{x}, \mathbf{x}_i) = \begin{cases} y_i & \text{if } d(\mathbf{x}, \mathbf{x}_i) \leq \rho \\ \overline{y_i} & \text{otherwise} \end{cases} \tag{4.1}$$

where, $\overline{y_i}$ denotes the Boolean complement of y_i, and $d(\mathbf{x}, \mathbf{x}_i)$ denotes the distance between these two examples. Equation (4.1) means that if the distance is equal to or less than some real value ρ, the feature $h_{i,\rho}$ outputs y_i, otherwise $\overline{y_i}$. When there is no ambiguity about ρ, we abbreviate $h_{i,\rho}(\mathbf{x})$ to $h_i(\mathbf{x})$.

The hypothesis $f(\mathbf{x})$ of the SCM is expressed as a logical formula. It is represented by conjunctions or disjunctions of features as follows:

$$f(\mathbf{x}) = \begin{cases} \bigwedge_{i \in \mathcal{R}} h_i(\mathbf{x}) & \text{for a conjunction} \\ \bigvee_{i \in \mathcal{R}} h_i(\mathbf{x}) & \text{for a disjunction} \end{cases} \tag{4.2}$$

For the conjunction case, $f(\mathbf{x}) = 1$, if $h_i(\mathbf{x}) = 1$ for all i, otherwise $f(\mathbf{x}) = 0$. For the disjunction case, $f(\mathbf{x}) = 0$, if $h_i(\mathbf{x}) = 0$ for all i, otherwise $f(\mathbf{x}) = 1$.

To generate an accurate hypothesis, useful features must be selected. In the SCM, the *usefulness* of a feature is defined to determine useful features. The usefulness U_h of feature h is defined as follows:

$$U_h \overset{\text{def}}{=} |Q_h| - p \cdot |R_h| \tag{4.3}$$

where, in a conjunction (disjunction) case, $|Q_h|$ denotes the number of negative (positive) examples that are correctly classified by h, and $|R_h|$ denotes the number of positive (negative) examples that are misclassified by h. The penalty value p is defined for the tradeoff between the accuracy of the training data and the complexity of the hypothesis. A hypothesis consists of the features that are selected in descending order of their usefulness.

4.3 Visual Learning with Set Covering Machine

The learning process of our method consists primarily of three components (see Figure 4.1). First, image filtering is conducted to extract useful features of images in order to generate an accurate hypothesis. There are 17 filters used in our method, consisting of Intel Image Processing Libraries [3] and OpenCV Libraries [4]. Second, there is the learning with the SCM to get a hypothesis. Third, the hypotheses generated by each filtering to find the best filter are evaluated. The learning proceeds by repeating these three operations D times, then finally outputting a final hypothesis. D is the number of filters applied to the training images. The parameter is given by the user. The final hypothesis $f_{\hat{D}}$ is the hypothesis which has the highest value of the evaluation function among the hypotheses generated during the learning loop (step 2 to step 5 in Figure 4.1). Thus, the final hypothesis $f_{\hat{D}}$ is given by:

$$f_{\hat{D}} = \underset{f \in \{f_1, \cdots, f_D\}}{\operatorname{argmax}} E(f) \tag{4.4}$$

where, E is the evaluation function (defined later).

```
1. Input training images, and $d \leftarrow 1$.
2. For all filters, obtain a hypothesis with SCM.
3. Evaluate each hypothesis using the evaluation function.
4. Retain the best hypothesis $f_d$ which has the highest
   value of the evaluation function.
5. **If** $d = D$, then goto step 6.
   **Otherwise**, apply the best filter to all the training images,
   $d \leftarrow d + 1$, and goto step 2.
6. Output the final hypothesis $f_{\hat{D}}$ which has
   the highest estimation value.
```

Figure 4.1: The learning process with the SCM.

Our method uses raw images as input data. An image consists of (*width* $\times$ *height*) pixels. Provided that an image is a gray-scale image, each pixel's depth is represented in bits. Its intensity is expressed by a scalar value. The intensity of a n-bit depth pixel ranges from 0 to $2^n - 1$. Thus, an example is represented as $(\mathbf{x}, y)$, where $\mathbf{x}$ is a vector of the intensity values and y is a class label.

We chose to construct a hypothesis with conjunctions of features for the following two reasons: first, it is reported in [19] and [20] that the difference between the conjunction and disjunction cases in the average classification accuracy is not significant. Second, the computational complexity is almost equal because the algorithms of the conjunction and disjunction cases are symmetrical. For our method, we define a feature $h_{i,j,\rho}$ by the distance between the intensity of the j-th pixel of two examples as follows:

$$h_{i,j,\rho}(\mathbf{x}) \stackrel{\text{def}}{=} h_\rho(\mathbf{x}_j, \mathbf{x}_{i,j}) = \begin{cases} y_i & \text{if } |\mathbf{x}_j - \mathbf{x}_{i,j}| \leq \rho \\ \overline{y_i} & \text{otherwise} \end{cases} \tag{4.5}$$

where, $\mathbf{x}_j$ denotes the intensity of the j-th pixel. We abbreviate $h_{i,j,\rho}(\mathbf{x})$ to $h_i(\mathbf{x})$ when there is no ambiguity about j and ρ.

We fixed the penalty parameter p in equation (4.3) to ∞ because this parameter setting simplifies the procedure for constructing the hypothesis. By this setting, the features that misclassify one or more positive examples are never selected. Thus, a hypothesis consists only of the features that correctly classify all the positive examples. In our method, a hypothesis correctly classifies all the training examples if the two examples that have the same intensity values for all pixels and have opposite class labels do not exist.

To make our method efficient, we address the following four points and discuss these points elaborately:

1. Find a small number of useful features to reduce the computational complexity.
2. Evaluate hypotheses to find the best filter to be applied.
3. Search the best sequence of filters efficiently.
4. Extend the algorithm to solve multi-class problems.

4.3.1 Feature Selection

In the case of the first point, we present the method to select useful features by evaluating the usefulness of each attribute. The number of possible features is mp, where m is the number of training examples and p is the number of attributes. The number of attributes is very large even if an image is not very large. However, the number of useful features is small compared with that of useless features. Thus, the computational complexity of finding useful features tends to be excessively high. Generally, as the computational complexity increases, the learning time becomes longer. Hence, the computational complexity can be reduced considerably by selecting useful attributes to reduce the number of possible features.

Figure 4.2 (a) is an example of a useful attribute. In this case, positive and negative examples are separable. Thus, all the examples are correctly classified by only one feature. In contrast, in the case of Figure 4.2 (b), it is impossible to classify all the examples correctly because there are two examples that have the same intensity values and opposite class labels. Moreover, it is obvious that even if these examples are eliminated, the remainder is not separable by one feature.

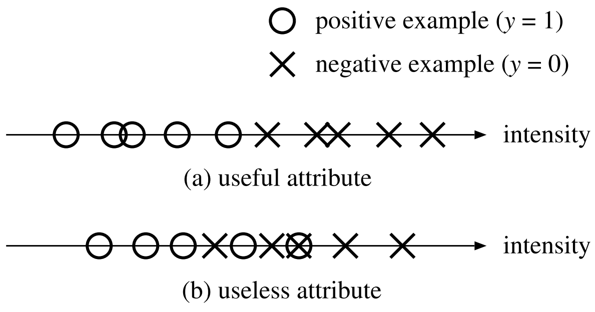


Figure 4.2: A useful attribute and a useless attribute.

We use the statistics of the training examples as the criterion of the usefulness of an attribute. If the variances of the intensity values of the positive and negative examples are small, and the distance between the mean intensity values of the positive and negative examples is large, then the two classes will be separable. Thus, we define the usefulness $u(i)$ of the i-th attribute as

$$u(i) = \frac{s^2(i)}{s^2_{\mathcal{P}}(i) + s^2_{\mathcal{N}}(i)} \tag{4.6}$$

where, $s^2(i)$, $s^2_{\mathcal{P}}(i)$ and $s^2_{\mathcal{N}}(i)$ are the variance of mean intensity of each class of the i-th attribute, and variances of the i-th attribute's intensity of the positive and negative examples respectively. That is,

$$s^2(i) = (\mu(i) - \mu_{\mathcal{P}}(i))^2 + (\mu(i) - \mu_{\mathcal{N}}(i))^2 \tag{4.7}$$

$$s^2_{\mathcal{P}}(i) = \frac{\sum_{\mathbf{x}\in\mathcal{P}}(x_i - \mu_{\mathcal{P}}(i))^2}{N_{\mathcal{P}}} \tag{4.8}$$

$$s^2_{\mathcal{N}}(i) = \frac{\sum_{\mathbf{x}\in\mathcal{N}}(x_i - \mu_{\mathcal{N}}(i))^2}{N_{\mathcal{N}}} \tag{4.9}$$

where,

x_i :	the i-th attribute's intensity of example $\mathbf{x}$.
$\mathcal{P}$ [$\mathcal{N}$] :	the set of positive [negative] examples.
$N_{\mathcal{P}}$ [$N_{\mathcal{N}}$] :	the number of positive [negative] examples.
$\mu_{\mathcal{P}}(i)$, $\mu_{\mathcal{N}}(i)$, and $\mu(i)$:	the mean value of the i-th attribute's intensity of positive, negative, and all examples, respectively.

In our method, if the number of possible features is large, only a few features that have a high usefulness are used to generate a hypothesis. The number of features M to be used is determined based on the number of the training images. The maximum number of features M_{max} included in the hypothesis of the SCM is given by

$$M_{max} = \max\{N_{\mathcal{N}},\ N_{\mathcal{P}} + 1\}. \tag{4.10}$$

Hence, M_{max} features are considered to be sufficient to generate a hypothesis and we define M as

$$M \overset{\text{def}}{=} \min\{M_{max},\ a\} \tag{4.11}$$

where, a is the number of attributes of the training images. The hypothesis $H(\mathbf{x})$ consists of M selected features $\eta_1(\mathbf{x}), \cdots, \eta_M(\mathbf{x}) \in \{h_1(\mathbf{x}), \cdots, h_{mp}(\mathbf{x})\}$, and is given by

$$H(\mathbf{x}) \quad = \quad \bigwedge_{i=1}^{M} \eta_i(\mathbf{x}). \tag{4.12}$$

4.3.2 Evaluation of Filters

For the second point, we define an evaluation function based on the information gain [28] to measure the usefulness of a filter. The information gain is a measurement to determine useful attributes in order to classify the training examples efficiently and is used for decision tree learning algorithms such as C4.5 [27]. Though the information gain is usually computed for an attribute, in our method, the information gain is computed to evaluate all features included in a hypothesis. Since the information gain is the difference between the entropy of the $(i-1)$-th feature and the weighted mean of the entropy of the i-th feature, we define the information gain for the i-th feature $(i = 1, 2, \cdots, M)$ that

$$Gain(h_i) \overset{\text{def}}{=} Entropy(h_{i-1}) - \frac{N - \nu_i}{N - \nu_{i-1}} Entropy(h_i) \tag{4.13}$$

where M denotes the number of generated features, and $Entropy(h_i)$ denotes the entropy of the set $(\mathcal{E} \setminus \mathcal{N}_i)$, where $\mathcal{E}$ is the training set and $\mathcal{N}_i$ is the set of negative examples that are correctly classified by features $h_1, \cdots, h_{i-1}$. The entropy $Entropy(h_i)$ for the i-th feature $(i = 0, 1, 2, \cdots, M)$ is given by:

$$Entropy(h_i) = -\frac{N_{\mathcal{P}}}{N - \nu_i} \log_2 \frac{N_{\mathcal{P}}}{N - \nu_i} - \frac{N_{\mathcal{N}} - \nu_i}{N - \nu_i} \log_2 \frac{N_{\mathcal{N}} - \nu_i}{N - \nu_i} \tag{4.14}$$

where,

N : the number of training examples.
n_i : the number of negative examples correctly classified by the i-th feature h_i.
ν_i : the sum of $n_1, \cdots, n_i$, that is $\sum_{j=1}^{i} n_j$, and ν_0 is defined to be 0.

We define the evaluation function E as the weighted sum of the information gain of all the features as follows:

$$E \stackrel{\text{def}}{=} \sum_{i=1}^{m} \frac{n_i}{N_{\mathcal{N}}} Gain(h_i). \tag{4.15}$$

4.3.3 Search for the Best Sequence of Filters

For the third point, we introduce a method to search efficiently for the best sequence of filters. The learning process of our method can be regarded as a search for the best sequence (or combination) of filters. There are n^D possible sequences of filters, and the search space is represented as a tree structure shown in Figure 4.3. In Figure 4.3, the nodes in i-th depth are the candidates for the i-th filter.

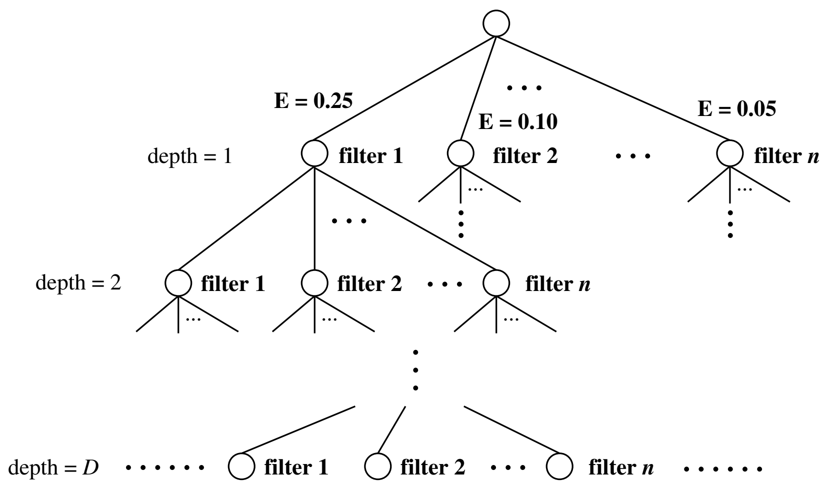


Figure 4.3: The tree structure of the search space.

The tree consists of $\sum_{i=0}^{D} n^i$ nodes including leaves, where n is the number of filters. Since the number of nodes in the tree increases exponentially as D increases, searching the overall nodes of the tree requires quite large amount of time.

Hence, we use the beam search (see Section 3.3), which is commonly used for efficient searching to reduce the computational complexity by restricting the search space (i.e. the number of nodes). The maximum number of nodes retained by the beam search is called *beam_width*. If the number of nodes to be retained is larger than the value of *beam_width*, then the beam search does not open the nodes which have low estimation values. If a certain node has a low estimation value, then its child nodes also seem to have low estimation values. Thus, by applying the beam search, the nodes that have low estimation values will not be opened. For example, in Figure 4.3, the node "filter 2" at depth 1, whose value of E is 0.10 and the node "filter n" at depth 1, whose value of E is 0.05 will not be opened.

4.3.4 Extension to Multi-class Problems

For the fourth point, we describe the method to solve multi-class problems. Since the SCM originally solves only binary (two-class) problems, when a classification problem contains more than two classes, we must extend the SCM to solve multi-class problems. To deal with multi-class problems, we used the *one-against-all* method. In the one-against-all method, k hypotheses are obtained by executing the learning algorithm k times, where k denotes the number of classes. The i-th hypothesis is generated by regarding all of the examples that belong to the i-th class as positive examples, and the remainder as negative examples.

4.4 Experiments

We performed some experiments using the synthetic aperture radar (SAR) data set in the MSTAR SAR database [10]. The learning task is to recognize the SAR images of the three different objects (i.e. classes): BMP-2 Armored Personnel Carrier, BTR-70 Armored Personnel Carrier, and T-72 Main Battle Tank. We used a set of 585 images which had been split into a disjoint training set and a test set. All images in the training set and the test set were regarded as gray scale (8-bit depth) images and were resized to 48 $\times$ 48 pixels.

We carried out some experiments, the result of which are shown in Figure 4.4. To confirm the effectiveness of feature selection and the beam search, we show the results with four different settings: with feature selection and the beam search (feature selection + search), with feature selection only (feature selection), with the beam search only (search), and with neither feature selection nor the beam search (none). We fixed the value of *beam_width* at 10.

Compared with the results produced without feature selection, the use of feature selection considerably shortens the learning time, whereas the classification accuracy is almost equal. By using feature selection, the number of features used to generate a hypothesis is reduced from 2304 to 98. It is obvious that useful features are selected by feature selection. With a small number of features, it is possible to generate a hypothesis which is as accurate as the hypothesis generated by using all the attributes. From the results, it can be said that the beam search is effective in improving the classification accuracy but that the

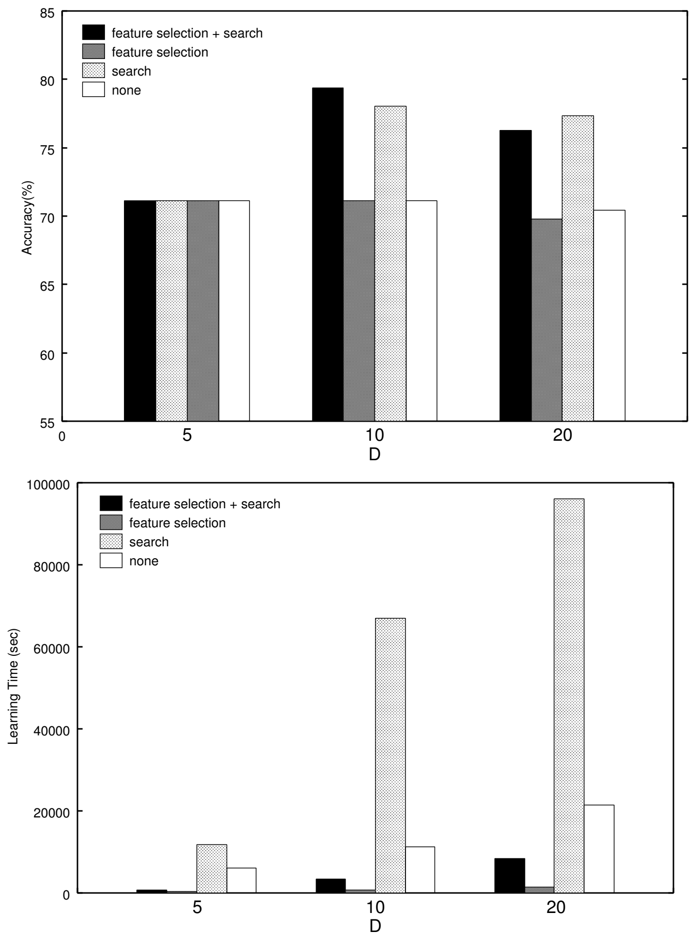


Figure 4.4: The results of experiments for the MSTAR SAR data set.

search requires a relatively long learning time. Provided that the beam search is carried out, the classification accuracy with $D = 10$ is fairly high compared with the classification accuracy with $D = 5$. However, the classification accuracy with $D = 20$ is almost equal to the classification accuracy with $D = 10$. This implies that to some extent the classification accuracy increases in proportion to the value of D. However, assigning too high a value to D will make the learning time quite long. Moreover, it will not further improve the classification accuracy of the hypothesis. Thus, we must take into consideration the tradeoff between the classification accuracy and the learning time. It is desirable to find some criterion to determine the optimal parameter setting.

4.5 Adaptive Learning Based on the MDL Principle

In our method, the SCM continues learning until there are no more negative training examples that can be correctly classified. A feature that correctly classifies many negative training examples generally makes a hypothesis accurate. In contrast, a feature that correctly classifies few negative training examples is generally useless to generate an accurate hypothesis.

Hence, we propose to introduce adaptive learning into our method. To put it concretely, in step 3 of Figure 4.1, the hypothesis is evaluated with a cost function. If the value of the cost function of the hypothesis is higher than that of the previous hypothesis, the SCM stops learning (i.e., goes to step 6). By introducing adaptive learning, an improvement of our method is expected.

We defined the cost function C based on the stochastic complexity of the MDL principle [17]. The MDL principle provides a criterion for the trade-off between the simplicity of the model and the model's fitness for the data. One is called *model description length* and the other is called *data description length.* On one hand, the simpler the model, the smaller the model description length. On the other hand, the better the model's fitness for the data, the smaller the data description length. In our case, the model description length increases in proportion to the number of features and training examples. The data description length is proportional to the rate of misclassification. The MDL principle asserts that the best model poses the minimum value of the sum of the model description length and the data description length. By introducing the MDL-based criterion into our method, we can improve our method by excluding useless features from the hypothesis.

The cost function $C(h_i)$ of the i-th feature is defined as

$$C(h_i) \stackrel{\text{def}}{=} MD(h_i) + DD(h_i) \tag{4.16}$$

where $MD(h_i)$ and $DD(h_i)$ are the model description length and the data description length, respectively, of the i-th feature defined as

$$MD(h_i) \stackrel{\text{def}}{=} \frac{i}{2}\log_2 N \tag{4.17}$$

$$DD(h_i) \stackrel{\text{def}}{=} \sum_{j=1}^{i} n_j \log_2 \frac{n_j}{N_{\mathcal{N}}} - (N_{\mathcal{N}} - \nu_i)\log_2 \frac{N_{\mathcal{N}} - \nu_i}{(\varepsilon - i)N_{\mathcal{N}}} \tag{4.18}$$

where, ε is the estimation of the number of features to be included in the hypothesis. Since the value of $f(i) = (N_{\mathcal{N}} - \nu_i)$ tends to decrease exponentially as i increases, we estimate the value by the function $f(i) = ae^{bi} - 1$. The coefficients a and b are determined by the two conditions: $f(0) = N_{\mathcal{N}}, f(1) = N_{\mathcal{N}} - n_1$. Thus, we defined ε experimentally as

$$\varepsilon \overset{\text{def}}{=} \frac{\ln(N_{\mathcal{N}}+1)}{\ln(N_{\mathcal{N}}+1)-\ln(N_{\mathcal{N}}-n_1+1)}. \tag{4.19}$$

Here, $\ln x$ denotes the natural logarithm of x, and $f(\varepsilon)=0$.

We performed some experiments using the same data set and the same parameter settings as in the experiments in Section 4.4. To verify whether adaptive learning improves the performance of our method, we show the results with four different settings: the results with adaptive learning and the beam search (adaptive + search), with adaptive learning only (adaptive), with the beam search only (search), and with neither adaptive learning nor the beam search (none). For all experiments, feature selection was carried out. The results are shown in Figure 4.5.

By introducing adaptive learning, we reduced the average number of features in the hypothesis from 27.7 to 13.3; thus, the learning time was also reduced by $20 \sim 30\%$. It is clear from the results that a reduction in the number of features is effective to reduce the computational complexity. However, the classification accuracy did not change for all experiments. This result implies that the features which correctly classify few negative training examples have little influence over the overall classification ability of the hypothesis. The parameter ε may be connected with the performance of adaptive learning, since the number of features included in the hypothesis decreases in proportion to the reduction of ε, while the value of ε is sometimes underestimated. More precise estimation of the value of ε may be effective in improving adaptive learning method. From the experimental results, it is implied that adaptive learning is effective in improving the computational complexity of our method.

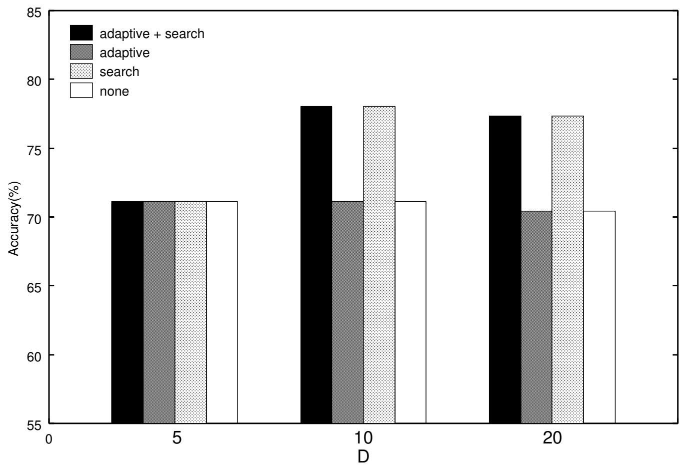
4.6 Conclusion

We have proposed an efficient visual learning method using feature selection based on the SCM algorithm. In order to find useful pixels, we introduced the idea of discriminant analysis and developed an effective criterion to evaluate the usefulness of each pixel. Consequently, the computational complexity is considerably shortened without the degradation of classification accuracy.

We applied image filtering to given images in order to extract useful features. Since the number of the combination of filters is exponential, we proposed an efficient method to determine an appropriate combination of filters using the beam search.

To judge the convergence of learning, we proposed an adaptive learning method based on the MDL principle. We have shown the effectiveness of the proposed method through several image classification tasks.

However, the proposed method is rather simple because it utilizes only a single type of feature (i.e., the intensity distribution of pixels). In order to distinguish real-world objects correctly, the utilization of diverse features seems to be more effective. In addition, although image filtering can remove slight noise



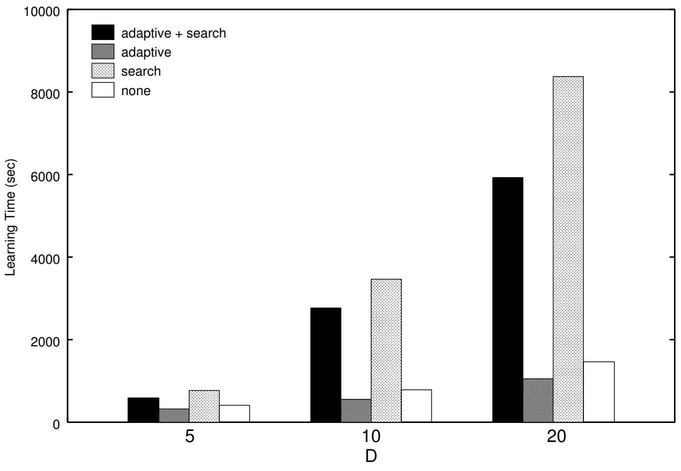


Figure 4.5: The results of experiments using adaptive learning.

in images, it will not be able to solve the problems of deformation, occlusion of objects, and cluttered backgrounds. Thus, we try to solve these problems by developing better visual learning. Details about this issue are described in the next chapter.

Chapter 5

Visual Learning with Collaboration among Diverse Classifiers

5.1 Introduction

In the previous chapter, we describe a visual learning method based on the SCM for real-world object recognition. Although this method performs flexible recognition by using the distribution of pixel intensity values, there is room for improvement in the flexibility and accuracy of recognition strategy because of the following problems.

The first problem stems from the features used to discriminate objects. Real-world objects are so complex that they cannot be correctly discriminated using a single feature, and conventional recognition methods including the visual learning method proposed in the previous chapter utilize a small number of features. To make effective use of diverse features, the idea of modularity should be introduced. A module corresponds to a simple recognition method that recognizes objects using a small number of features. Then, multiple modules are integrated to utilize a wide variety of features. This idea is based on the assumption that human visual systems achieve excellent recognition through the modularization of visual information processing [22]. There is, however, a crucial problem in that, although useful modules contribute to the improvement of recognition performance, useless modules deteriorate recognition performance and increase computational complexity. Useful modules are difficult to find because useful features depend on images. In addition, the problem of finding the optimal recognition model (i.e., the optimal combination of modules) is complicated by the large number of possible combinations. An efficient model selection method is, therefore, essential.

The second problem lies in given images. A training set, the images input into a recognition method, often includes objects that are deformed, appear in bad lighting conditions, etc. To deal with these images, transform algorithms such as the Fourier transform are introduced to compensate for the deformation of objects, while problems relating to lighting conditions are resolved using histogram modification [2]. It is extremely difficult, however, to resolve all of these problems for a variety of environments. While the human visual system has the intrinsic ability to correct for the deformations or poor lighting conditions of objects, it may fail to discriminate objects when they are in bad environments because the fundamental functions of human visual systems are built up in their infancy using comprehensible objects that are not deformed or in poor lighting conditions. This is a key point for the efficiency of human visual ability. Thus, we have to select appropriate images that contribute to improving recognition performance. Evaluating the appropriateness of an image is, however, a difficult task.

The third problem is the optimization of the recognition model. Because of the efficiency of modularity, several recognition methods based on modularity have been proposed [13] [5]. These methods, called multiple classifier systems, construct multiple modules by combining several features and/or image classification algorithms. The outputs of multiple modules are integrated using a ranking algorithm, a voting algorithm, etc. The problem is that these methods do not consider the interaction among modules but instead discriminate objects by simply combining the predictions of each module. This strategy is not efficient at recognizing complex objects because it does not intrinsically integrate diverse features. In order to take advantage of modularity, the recognition model should be optimized by accounting for the interaction among multiple modules. In particular, the modules should cooperate so that all modules complement each other.

In order to solve these problems, we propose an effective recognition method based on *visual learning*. Visual learning adopts the idea of machine learning into computer vision understanding to develop more flexible and accurate recognition methods [15].

First, we introduce an effective machine learning framework called *ensemble learning* [8] to solve the first problem. This framework constructs and integrates multiple modules called classifiers to make use of diverse features. Since the number of possible combinations of classifiers is quite large, we have to select the optimal one. Thus, we evaluate each classifier by defining the reliability of a classifier as the possibility that it correctly classifies the given example. Reliability is determined for each class (i.e., category of objects) and calculated using the classification accuracy for training examples. For example, assuming that a classifier X discriminates an image by analyzing its contours, X will correctly distinguish balls from dice because they have different contours. But it may fail to distinguish balls from oranges because they have similar contours. Thus, the reliability for dice is high, while that for balls and oranges is low. When an example is given, we select the optimal classifier based on reliability.

Next, we detect and deal with *hard examples* to solve the second problem. A hard example refers to an image that is hard to classify correctly. For example, deformed images and occluded images are hard examples. The problem is that hard examples cause overfitting. Overfitting means that a classifier is too specialized for the classification of hard examples and often misclassifies non-hard examples. Thus, hard examples should be eliminated. It is difficult, however, to detect hard examples appropriately. To resolve this issue, we propose an efficient learning strategy that detects hard examples by giving a weight to each example. The weight is a measure of the difficulty of correctly classifying the example and is increased when the example is misclassified and decreased when it is correctly classified. If the weight of an example becomes higher than a threshold, we regard the example as a hard example and eliminate it from the training set. This strategy reduces the influence of hard examples and thus prevents overfitting.

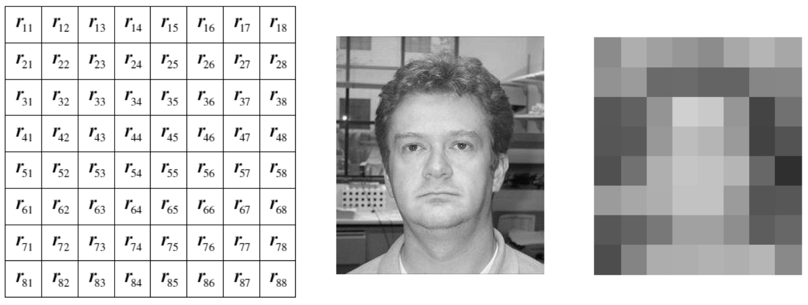
Finally, to solve the third problem, we introduce interaction among multiple classifiers by collaboratively training them so that they complement each other. The collaboration of classifiers is based on the following viewpoint: a hard example of a classifier can be correctly classified by other classifiers. According to this viewpoint, we introduce *collaboration of multiple classifiers*. Specifically, the classifiers are collaboratively trained as follows: when it turns out that the classifier X cannot correctly classify some examples, they are regarded as hard examples for X. Then, the hard examples are sent to other classifiers and, if these classifiers can correctly classify the examples, are used to train them. Conversely, the hard examples of other classifiers are sent to X and used for training if X can correctly classify them. As a result of this collaboration, the recognition model can be optimized during the learning process.

5.2 Collaborative Visual Learning Model

In order to deal with diverse features, we introduce the idea of ensemble learning. As mentioned in Chapter 2, however, although the recognition performance of separated ensemble models is higher than that of non-ensemble models, it is still inadequate because of the influence of hard examples. We show an example using two types of single-feature classifiers. The recognition task is to distinguish face images from non-face images. The first classifier uses as a feature two-dimensional histograms of pixel intensity, a simplified version of the face space in [37]. We show an example of intensity histograms in Figure 5.1.

Figure 5.1 (a) represents the two-dimensional intensity histogram. The histogram is obtained by dividing an image into 8×8 blocks (i.e., $r_{11}, r_{12}, \cdots r_{88}$) and calculating the average intensity value for each block. (b) and (c) are the original image (i.e., the training example) and its intensity histogram, respectively. The central blocks around the face have high values because of the brightness of skin. Conversely, several blocks around the hair have low values. The spatial relationship between bright and dark blocks represents the structure of the face.

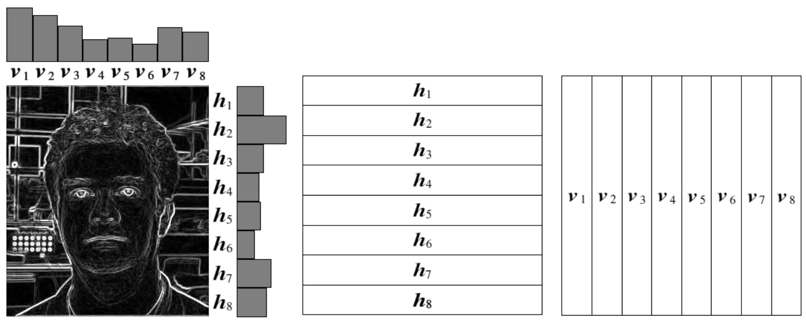
The second classifier is based on contour density histograms, which are obtained by extracting contours that represent the complexity of shapes in an


r_{11} r_{12} r_{13} r_{14} r_{15} r_{16} r_{17} r_{18}
r_{21} r_{22} r_{23} r_{24} r_{25} r_{26} r_{27} r_{28}
r_{31} r_{32} r_{33} r_{34} r_{35} r_{36} r_{37} r_{38}
r_{41} r_{42} r_{43} r_{44} r_{45} r_{46} r_{47} r_{48}
r_{51} r_{52} r_{53} r_{54} r_{55} r_{56} r_{57} r_{58}
r_{61} r_{62} r_{63} r_{64} r_{65} r_{66} r_{67} r_{68}
r_{71} r_{72} r_{73} r_{74} r_{75} r_{76} r_{77} r_{78}
r_{81} r_{82} r_{83} r_{84} r_{85} r_{86} r_{87} r_{88}

(a) two-dimensional histogram (b) original image (c) intensity histogram

Figure 5.1: The original images of the examples and its intensity histogram.

image. We show an example of contour density in Figure 5.2.

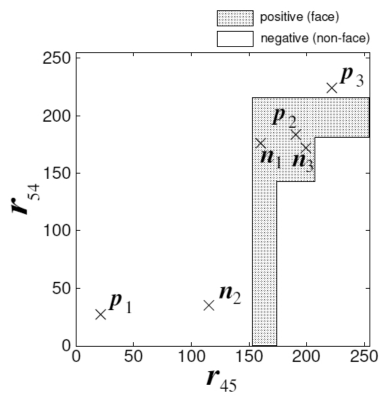

v_1 v_2 v_3 v_4 v_5 v_6 v_7 v_8
h_1 h_2 h_3 h_4 h_5 h_6 h_7 h_8
h_1 h_2 h_3 h_4 h_5 h_6 h_7 h_8
v_1 v_2 v_3 v_4 v_5 v_6 v_7 v_8

(a) contour density histogram (b) horizontal blocks (c) vertical blocks

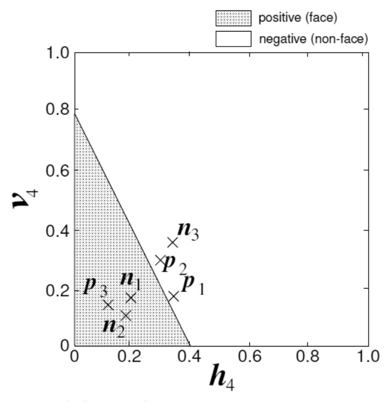
Figure 5.2: An example of contour density.

In a face image as depicted in Figure 5.2 (a), the contour density of the regions of eyes, nose and mouth tend to be high because they have relatively complex shapes. On the other hand, the forehead and cheeks have low contour densities because of their simple shapes. A contour density histogram is obtained through the following steps: First, the contours of objects are extracted from an image. Next, the contour image is divided into 8 horizontal blocks and 8 vertical blocks as described in (b) and (c). Finally, the contour density for each block ($h_1, \cdots, h_8$ and $v_1, \cdots, v_8$) is calculated as the ratio of the contours to the whole image. The horizontal and vertical contour histograms are shown in (a).

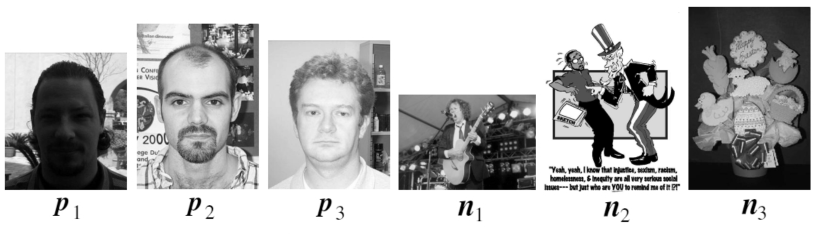
In ensemble learning, hard examples are divided into two types. To show this, we trained two classifiers X_I and X_C using approximately 400 face images as positive examples and 400 non-face images as negative examples. X_I is trained using intensity histograms, and X_C is trained using contour density histograms. We used the intensity histograms of only two blocks r_{45} and r_{54}, which are located in the center of a face. Similarly, we use the contour density histograms of only two blocks h_4 and v_4 because they are also located in the center of a face. We visualize X_I and X_C in Figure 5.3.



(a) X_I (trained using intensity histogram)



(b) X_C (trained using contour density histogram)



(c) corresponding images

Figure 5.3: The learning results of X_I and X_C and the images of some hard examples.

Figure 5.3 (a) and (b) represent the learning results of X_I and X_C respectively. The positive examples p_i and the negative examples n_i $(i = 1, 2, 3)$ are the hard examples. p_1 and n_1 are misclassified by both X_I and X_C because of the bad lighting condition of p_1, and a person and cluttered background in n_1. We call these examples *true hard examples*, which refers to examples misclassified by all classifiers. p_2 and n_2 are correctly classified by X_I but misclassified by X_C. p_2 is misclassified due to the complex background. n_2 is misclassified because it contains an illustration of persons whose contours are partially similar to those of human faces. We call these examples *feature-dependent hard examples*, which refers to examples misclassified by some classifiers but correctly classified by other classifiers. Conversely, p_3 and n_3 are correctly classified by X_C and misclassified by X_I. p_3 is misclassified because its light condition is too bright. The misclassification of n_3 is caused by its intensity distribution which is similar to that of a face. These are also feature-dependent hard examples.

True hard examples should be eliminated to avoid overfitting, while feature-dependent hard examples should be utilized to train classifiers that are able to classify them correctly. Thus, p_2 and n_2 should be used to train X_I, and p_3 and n_3 should be used to train X_C. Separated ensemble models, however, cannot make use of these examples because they are not able to distinguish feature-dependent hard examples from true hard examples. This leads to crucial learning inefficiency.

In order to solve this problem, we propose a collaborative ensemble model based on the following viewpoints:

- Introduce interaction between classifiers by collaboratively training each classifier.
- Eliminate true hard examples from all training sets.
- Utilize feature-dependent hard examples for each classifier if the other classifiers can classify them correctly.

The structure of collaborative ensemble model is represented in Figure 5.4.

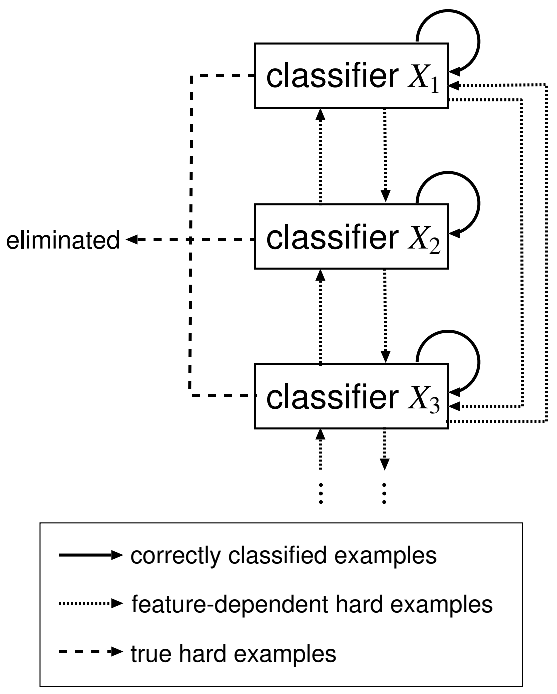


Figure 5.4: The collaborative ensemble model.

In the collaborative ensemble model, an arbitrary number of classifiers can be simultaneously trained. To detect feature-dependent and true hard examples, each classifier X_i ($i = 1, 2, \cdots, n$ where n is the number of classifiers) informs the other classifiers about examples misclassified by X_i. Similarly, the other classifiers inform X_i about the examples they have misclassified. In this manner, X_i is trained using the feature-dependent hard examples of the other classifiers and vice versa. In addition, examples that are misclassified by all the classifiers are regarded as true hard examples and eliminated to prevent overfitting. The collaboration between classifiers enables the exploitation of training examples and, thereby, learning that is more efficient than separated ensemble models.

5.3 Collaborative Visual Learning Algorithm

5.3.1 A Weighting Algorithm to Detect Hard Examples

To describe the proposed algorithm, we first formulate an object recognition task as a classification problem. The training set S is represented as $S = \{(x_1, y_1), \cdots, (x_m, y_m)\}$, where m is the number of training examples, and x_i and $y_i \in \{1, \cdots, C\}$ correspond respectively to an image and a class label for recognition of C kinds of objects. Each example has a *weight* that measures the difficulty of correctly classifying the example. We iteratively train a classifier using the weights. Referring to an iteration as a round, the weights of all examples are updated to assess their classification difficulty at the end of each round. A higher weight means that the example is more difficult to classify correctly. In the machine learning domain, it has been proven that training a classifier using examples with high weights leads to an improvement in the performance of the classifier unless the examples are hard examples [8].

In order to train each classifier using examples that have high weights, in each round, we construct a training set by sampling examples from the training set S according to their weights. In AdaBoost [11], this process is referred to as a resampling of training examples. This method only works well, however, under the assumption that a training set contains no (or few) hard examples. If hard examples are included in the training set, overfitting occurs because the weighting algorithm of AdaBoost gives excessively high weights to hard examples. To resolve this issue, we propose a weighting algorithm that uses a weight distribution. A higher value for the weight distribution means that further use of the example will improve the performance of a classifier. We propose a weighting algorithm based on the following two points: (1) To prevent overfitting, when an example is regarded as a hard example by a classifier X, it should not be sampled as a training example for X in subsequent rounds. (2) To collaboratively train each classifier, when X misclassifies a training example, its weight distribution for other classifiers should be increased so that the example is used to train them. We, therefore, define the weight distribution $D^l_{i,t}$ of the i-th example x_i for the l-th classifier X^l in the t-th round as follows:

$$D^l_{i,t} = \frac{\delta^l_{i,t} d^l_{i,t}}{\sum_{i=1}^{m} \delta^l_{i,t} d^l_{i,t}} \tag{5.1}$$

where $\delta^l_{i,t}$ is 0 if x_i is regarded as a hard example by X^l (i.e., the weight of x_i becomes higher than the threshold[1]), otherwise $\delta^l_{i,t}$ is 1, and

$$d^l_{i,t} = \frac{1}{n-1} \sum_{j \neq l}^{n} \left(\frac{w^j_{i,t}}{\sum_{i=1}^{m} w^j_{i,t}} \right), \tag{5.2}$$

where n is the number of classifiers and $w^j_{i,t}$ is the weight of x_i for X^l in the t-th round. When an example x_i is regarded by X^l as a hard example, $\delta^l_{i,t} = 0$. Thus, from equation (5.1), $D^l_{i,t}$ is 0 and x_i is never sampled in any subsequent rounds. Equation (5.2) represents the collaboration of classifiers. $d^l_{i,t}$ depends on the weights of the examples of the other classifiers. Thus, the weight distribution of examples misclassified by other classifiers tends to be high. Consequently, this weighting algorithm works well even if many hard examples are included in the training set.

5.3.2 Construction of an Ensemble Classifier Using Collaboration among Diverse Classifiers

Assuming that the number of rounds is T, the l-th classifier trained in the t-th round is represented as X^l_t ($t = 1, \cdots, T$). At the end of the T-th round, we integrate the classifiers $\{X^l_t\}^T_{t=1}$ into an ensemble classifier X^l, and determine the prediction of X^l by integrating the predictions of all classifiers $\{X^l_t\}$ using a voting method. That is, the class label predicted by the ensemble classifier corresponds to the class label which is predicted by the majority of classifiers. In order to take the classification performance of each classifier into consideration, we introduce weighted voting. The prediction $X^l(x)$ of the ensemble classifier X^l for an example x is defined as follows:

$$X^l(x) = \operatorname*{argmax}_{c \in \{1, \cdots, C\}} \sum_{t=1}^{T} \alpha^l_t [X^l_t(x) = c], \tag{5.3}$$

where $[X^l_t(x) = c]$ is 1 if $X^l_t(x) = c$ and otherwise is 0. $\alpha^l_t = \log \frac{1-\varepsilon^l_t}{\varepsilon^l_t}$, where ε^l_t is the classification error of X^l. Thus, a higher value for α^l_t means a lower classification error.

In the learning process, each classifier is specialized to classify non-hard examples precisely. Thus, we should determine whether the given example is a hard example or a non-hard example. To distinguish hard examples from non-hard examples, we define a criterion and call it class separability. Class separability is defined so that it is proportional to the classification performance of a classifier for each class. When X^l_t correctly classifies most of the examples whose class labels are c, the class separability for class c is high. If X^l_t misclassifies many examples into the class c, the class separability for class c is low.

[1]The threshold is determined by searching for the optimal value using the beam search, whose algorithm is shown in Section 3.3.

Since hard examples are frequently misclassified, the class separability of a hard example will be much lower than that of a non-hard example. Here, we consider the case where X_t^l predicts the class label of an unseen example x as class c. If the class separability of X_t^l for class c is high, x will be a non-hard example. That is, the possibility that the prediction is correct will be high. We define the class separability $s_t^l(c)$ for class c as follows:

$$s_t^l(c) = \begin{cases} s_{t+}^l(c) & \text{if } X_t'^l(x) = c, \\ s_{t-}^l(c) & \text{otherwise,} \end{cases} \tag{5.4}$$

where

$$X_t'^l(x) = \underset{c \in \{1,\cdots,C\}}{\text{argmax}} \sum_{\tau=1}^{t} \alpha_\tau^l [X_\tau^l(x) = c],$$

$$s_{t+}^l(c) = \frac{n_{t,c,c}^l}{\sum_{i=1}^C n_{t,c,i}^l}, \quad s_{t-}^l(c) = \frac{\sum_{i \neq c}^C \sum_{j \neq c}^C n_{t,i,j}^l}{\sum_{i \neq c}^C \sum_{j=1}^C n_{t,i,j}^l}.$$

$n_{t,i,j}^l$ denotes the number of examples whose class label is i but which were misclassified into class j. $s_{t+}^l(c)$ is also high when many examples with class label c are correctly classified into class c. $s_{t-}^l(c)$ is high when many examples with a class label other than c are classified into a class other than c. If the prediction $X_t'^l(x)$ of the l-th ensemble classifier is c, $s_{t+}^l(c)$ is used as the class separability for class c because $s_{t+}^l(c)$ represents the classification accuracy for the class c. On the other hand, if the prediction $X_t'^l(x)$ is not c, $s_{t-}^l(c)$ is used as the class separability for the class c because $s_{t-}^l(c)$ corresponds to the recognition accuracy for all classes other than c.

When classifying an unseen example, we first obtain the predictions of all ensemble classifiers $\{X^l\}_{l=1}^n$, where n is the number of features. We next calculate the class separability of each ensemble classifier and select the classifier with the highest class separability as the most reliable classifier. Then, the final prediction $F(x)$ for an example x is determined by the following equation:

$$F(x) = X^{l^*}(x) \text{ such that } l^* = \underset{l}{\text{argmax}} \sum_{\tau=1}^{t} s_\tau^l(X_\tau'^l(x)). \tag{5.5}$$

Finally, we show the algorithm list of the collaborative ensemble method as follows:

1. Initialize: $t = 1$, $D_{i,1}^l = \frac{1}{m}$ and $\delta_{i,1}^l = 1$ for all i and l.

2. For each classifier, construct the training set S_t^l by sampling from the original training set S according to the weight distribution $D_{i,t}^l$.

3. Train each classifier using S_t^l and obtain X_t^l.

4. If $t = T$ then make the final prediction F and finish the learning process, otherwise go to step **5**.

5. For all l, classify all training examples using X_t^l. Then, decrease the weights of correctly classified examples and increase that of misclassified examples.

6. Determine weight thresholds for each classifier to detect hard examples.

7. Set $\delta_{i,t+1}^l$ to 0 if x_i is regarded as a hard example by X_t^l, otherwise 1.

8. Calculate the weight distribution $D_{i,t+1}^l$ for each l and i.

9. $t \leftarrow t+1$ and go to step **2**.

5.4 Experiments

We carried out several experiments to verify the performance of our method using the images in the ETH-80 Image Set database [16]. This data set contains 8 different objects: apples, cars, cows, cups, dogs, horses, pears, and tomatoes. We used 20% of the examples as training examples and the remainder as test examples. The number of rounds was experimentally set to 100. We constructed five types of classifiers using five types of features as given in Figure 5.5.

The first classifier is an appearance-based recognition method which utilizes contour fragments [25], as shown in (b). The set of contour fragments is grouped into meaningful structures called patterns as provided in Figure 5.5 (c). The second classifier is based on the distributions of pixel intensity values [25]. The distributions are represented by a Generic Fourier Descriptor (GFD). A GFD is obtained by calculating the spatial frequency of an image as described in (d). The third classifier is a feature tree [24]. This method generates region-based features by image filtering as shown in (e). It then combines several features into several decision trees called feature trees. The fourth classifier is based on Scale Invariant Feature Transform (SIFT) [18]. SIFT generates deformation-invariant descriptors by finding some distinctive points in objects (indicated by white arrows in (f)). The points represent the characteristics of the object based on image gradients. The fifth classifier uses the shape context method [1]. In this method, the contour of an object is described by a set of points as illustrated in (g). Using the set of points, log-polar histograms of the distance and angle between two arbitrary points, called shape contexts, are calculated for all points. An example of shape context is given in (h). This method discriminates the object by matching its shape contexts with the shape contexts in the training set.

5.4.1 Collaborative Model vs Non-Collaborative Model

To verify the effectiveness of our collaborative ensemble model, we construct four types of classifiers, L_2, L_3, L_4 and L_5, by integrating two, three, four and five classifiers respectively. L_i consists of the first, second, $\cdots$, and i-th classifiers. Then, we compare their performance. The result of the experiment is provided in Figure 5.6.

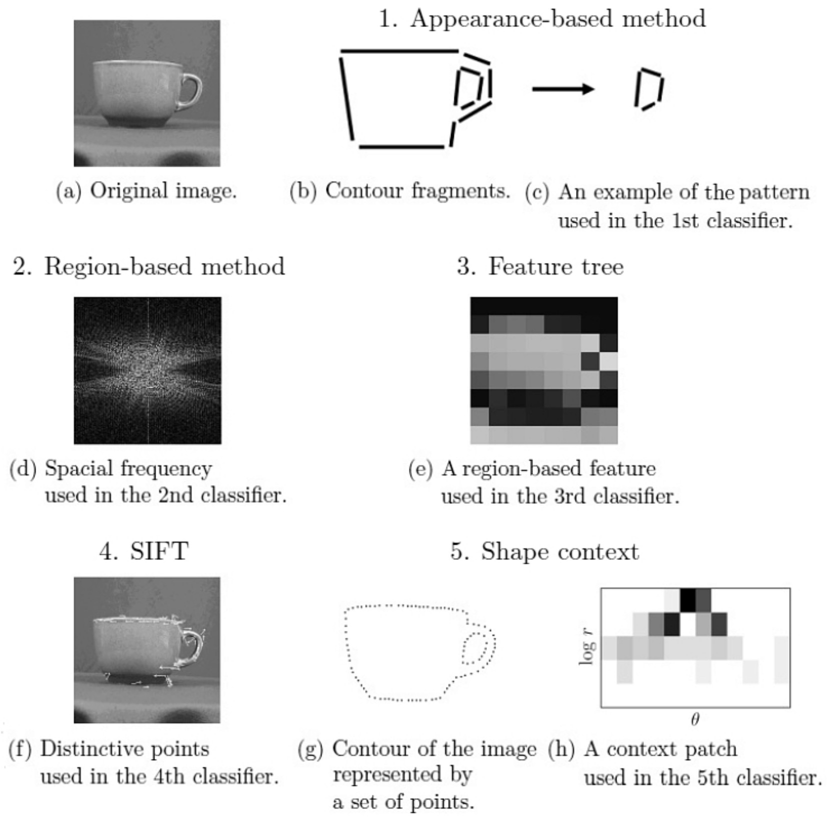

1. Appearance-based method
(a) Original image.
(b) Contour fragments.
(c) An example of the pattern used in the 1st classifier.
2. Region-based method
3. Feature tree
(d) Spacial frequency used in the 2nd classifier.
(e) A region-based feature used in the 3rd classifier.
4. SIFT
5. Shape context
log r
θ
(f) Distinctive points used in the 4th classifier.
(g) Contour of the image represented by a set of points.
(h) A context patch used in the 5th classifier.

Figure 5.5: The features used in this experiment.

Recognition accuracy is proportional to the number of integrated classifiers. In particular, accuracy improves significantly for animals with complex shapes and textures. This result implies that diverse features are required to discriminate correctly between complex objects and that our method can effectively utilize various features.

L_5 achieved much higher accuracy than the other classifiers. Although this improvement is due to the high classification performance of the shape context classifier (i.e. the fifth classifier), our method fully outperforms the shape context method. Since the shape context method depends on the shapes of objects, it sometimes fails to distinguish between objects which have similar shapes such as apples and tomatoes. Therefore, our method selects an appropriate classifier other than the shape context classifier for the classification of apples and tomatoes. This leads to a higher recognition accuracy for our method.

Next, we compare the recognition performance of the separated ensemble model with our collaborative ensemble model. In the separated ensemble model,

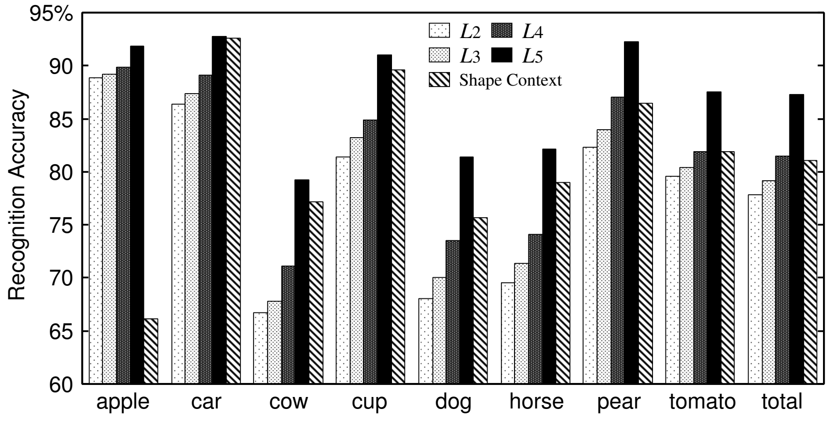


Figure 5.6: The recognition accuracy for each object classifiers.

each classifier is separately trained using AdaBoost. We construct four types of classifiers Λ_2, Λ_3, Λ_4 and Λ_5. Λ_i consists of the same classifiers as L_i. The recognition accuracy of each classifier is shown in Figure 5.7.

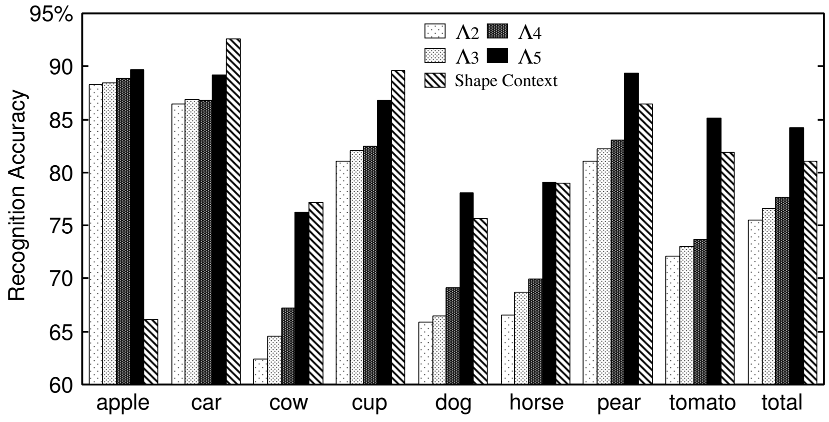


Figure 5.7: The recognition accuracy for the separated ensemble method.

For some objects, the classification accuracy is lower than that of the shape context method because the separated ensemble method is easily affected by hard examples. That is, if true hard examples are used as training examples, recognition performance is degraded. Moreover, since the separated ensemble method cannot exploit feature-dependent hard examples, learning efficiency is lower than for our method. This result, therefore, reflects the advantage of our model over the separated ensemble model.

5.4.2 The Combination of Classifiers

In our model, the combination of modules will affect its performance. Thus, recognition performance is dependent on the combination of classifiers. In order to examine this, we arbitrarily combined two classifiers and verified recognition performance. We show the recognition accuracy for each pair of classifiers in Figure 5.8.

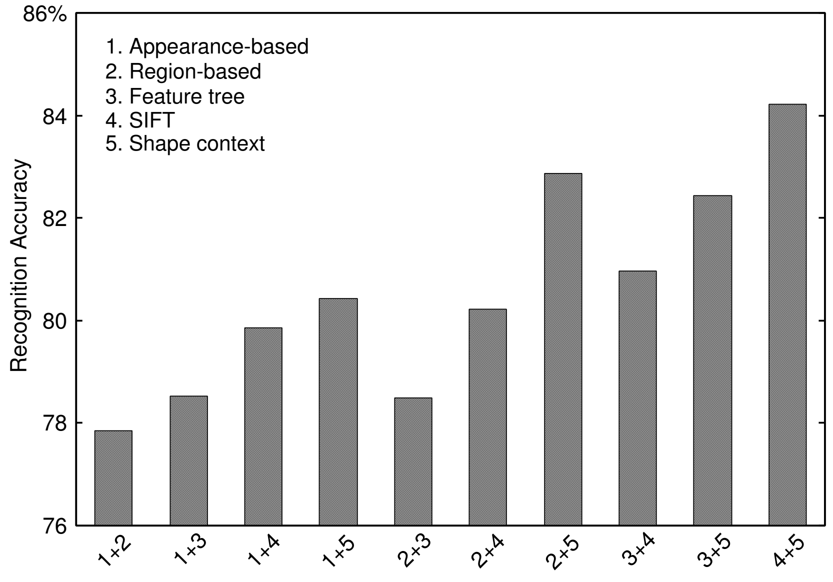


Figure 5.8: The recognition accuracy of each pair of classifiers.

For example, combining the second and fifth (Region-based and Shape context) classifiers is more effective than combining the first and fifth (Appearance-based and Shape context) classifiers. The features used for the second classifier are quite different from those used in the fifth classifier while both the first and fifth classifiers use the contours of objects as features. Similarly, as the fourth and fifth (SIFT and Shape context) classifiers are based on different features, combining them improves recognition performance. This result confirms that using diverse features leads to better model optimization.

Since finding the optimal combination of classifiers is an open problem, we determine the combination of classifiers experimentally. Our ensemble algorithm, however, enables an ensemble classifier to be less sensitive to the combination of classifiers because it determines the optimal classifier for a given example. To verify this, we performed an experiment using two types of classifier combination (we call them combination A and B) as described in Table 5.4.2.

For example, when the number of classifiers is 2, the first and second classi-

Table 5.1: The classifier combinations.

Number of classifiers	Used classifiers	
	Combination A	Combination B
2	1, 2	4, 5
3	1, 2, 3	3, 4, 5
4	1, 2, 3, 4	2, 3, 4, 5
5	1, 2, 3, 4, 5	1, 2, 3, 4, 5

fiers are used in combination A. In combination B, the fourth and fifth classifiers are used. For comparison, we show the recognition performance of a voting algorithm [8]. The result of the experiment is shown in Figure 5.9.

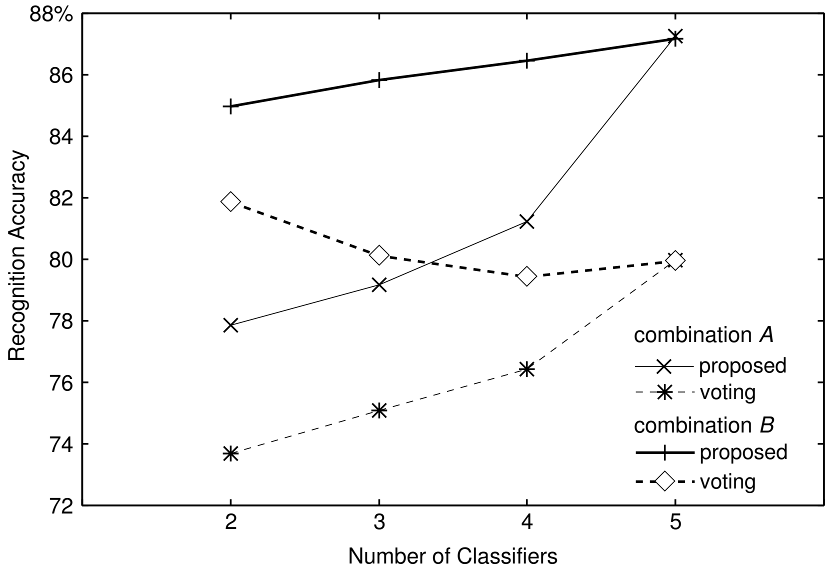


Figure 5.9: The recognition accuracy for each combination.

The recognition accuracy is proportional to the number of classifiers for combination A (denoted by $\times$ and $*$ in Figure 5.9 respectively). This is quite a reasonable result. For combination B, however, the recognition accuracy of the voting (denoted by $\diamond$) is degraded as the number of classifiers increases while our recognition accuracy (denoted by $+$) increases. This occurs because the voting treats the predictions of all classifiers equally regardless of individual recognition performance. Since the fourth and fifth classifiers have higher recognition performance than the first and second classifiers, in combination B, the recognition accuracy of the voting deteriorates when a classifier with a low recognition performance is added. On the other hand, we are able to avoid the degradation of recognition accuracy as a result of the model selection and optimization strategy, leading to stability with respect to differences in classifier combinations.

5.4.3 Comparison with Other Object Recognition Methods

We compare our recognition performance with that of the following six object recognition methods. The first is the shape context [1]. The second is the multidimensional receptive histogram [34], which describes the shapes of objects using statistical representations. The third is color indexing [35], which discriminates an object using RGB histograms calculated from all the pixels in the object. The fourth is based on local invariant features [12] that are generated by a gradient-based descriptor and are robust to the deformation of images. The fifth is the learning-based recognition method [21], where an object is described by randomly extracted multi-scale subwindows in the image and classified by an ensemble of decision trees. The sixth is the boosting-based recognition method [36], where a probabilistic boosting-tree framework is introduced to construct discriminative models. The recognition accuracy is shown in Table 5.2.

Table 5.2: The recognition accuracy for each object recognition methods (in %).

Swain [35]	64.85
Marée [21]	74.51
Tu [36]	76
Schiele [34]	79.79
Grauman [12]	81
Belongie [1]	86.40[2]
proposed	**87.27**

Our recognition accuracy is higher than that of each other recognition method. Since we utilize multiple features for recognition, we can thus discriminate a wider variety of objects than single-feature recognition methods. In addition, this result indicates that our learning strategy for selecting optimal classifiers works well. Since the criterion for the determination of optimal classifiers is determined by observing the collaborative learning process, this result implies the effectiveness of our collaborative learning framework.

[2] [16] reports that the recognition accuracy of the shape context method by Belongie et al. is 86.40%. This accuracy has been achieved, however, using over 98% of the examples in the training set while we use only 20%. In addition, its recognition accuracy is proportional to the number of the training examples as shown in [1]. The recognition accuracy of this method using 20% of examples in the training set is 81.06%.

5.4.4 Robustness to Hard Examples

In order to verify the robustness to hard examples, we carry out an experiment using the Caltech image data set, which contains many hard examples. We use the images of six kinds of objects from the data set: airplanes, cars, Dalmatians, faces, leopards and motorbikes. We use 20% of the examples as training examples and the remainder as test examples. We compare our recognition performance with that of the following two types of object classifiers. The first classifier, X_1, does not eliminate any hard examples and never increases the weights of hard examples even if they are misclassified. The second classifier, X_2, also does not eliminate hard examples, but it does increase the weights of all misclassified examples even if they are hard examples. First, we show the recognition accuracy for each class and total recognition accuracy in Figure 5.10.

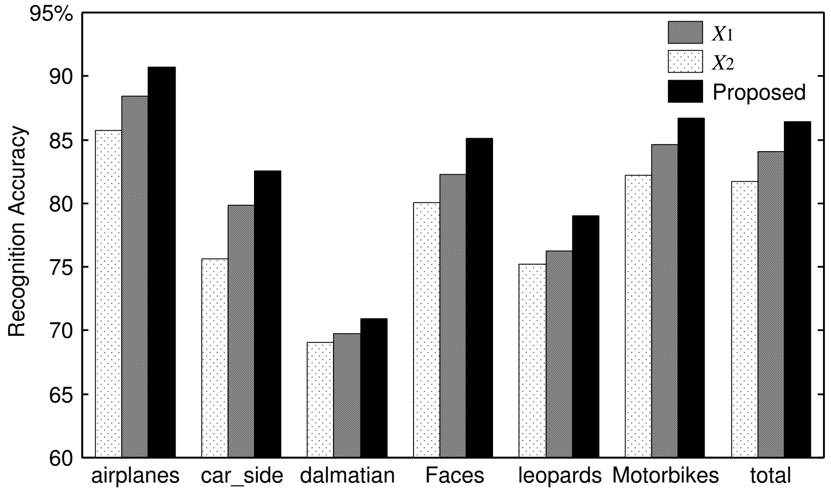


Figure 5.10: The recognition accuracy for each class.

The recognition accuracy of X_2 is the worst because it does not take hard examples into consideration. X_1 outperforms X_2 by giving smaller weights to hard examples and thus reducing their influence to some extent. Given that our recognition accuracy is the best for all objects, it seems that detecting and eliminating hard examples from training examples leads to more efficient learning.

Next, we show the transition of total recognition accuracy during the learning process in Figure 5.11. The recognition accuracy of X_2 decreases in the 70th round while the recognition accuracies of the other classifiers continue to increase as the number of rounds increases. This shows the sensitivity of X_2 to hard examples. Compared with X_1, our recognition accuracy seems to converge at an earlier round. This occurs because we can decrease the number of hard examples as learning proceeds. Thus, classifiers are trained without the influence of hard examples in later rounds. Consequently, we achieve a higher recognition performance and earlier learning convergence.

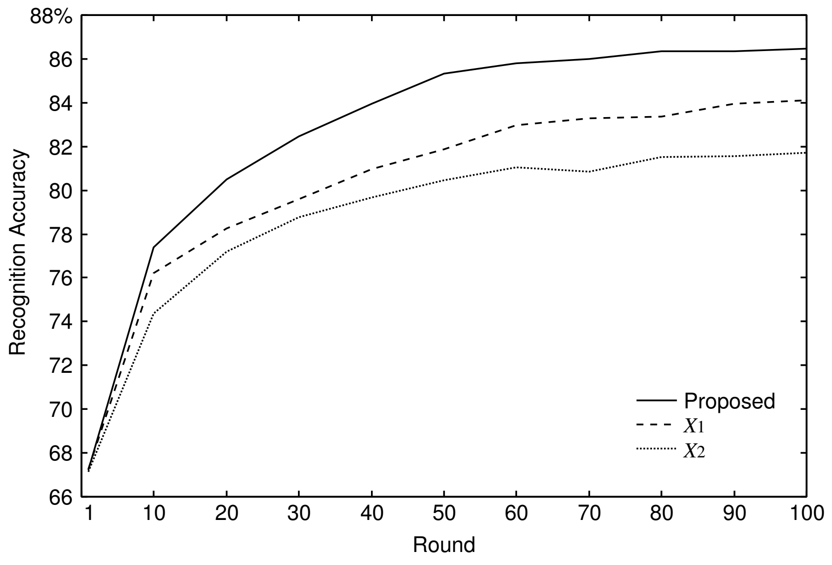


Figure 5.11: The transition of total recognition accuracy.

5.5 Conclusion

We have proposed an ensemble-based visual learning method using diverse features. In order to reduce the influence of hard examples and improve learning efficiency, we introduced interaction among multiple classifiers and proposed a collaborative ensemble learning model. We extended NadaBoost's algorithm to apply to multiclass problems and ensemble learning. We confirmed through several image recognition experiments that this led to robustness to hard examples. The proposed method can be made more accurate if we can determine the optimal combination of classifiers. Thus, we should find an efficient method to find this optimal combination. In addition, the computational complexity of the proposed method should be diminished by improving the collaborative learning model.

Chapter 6

Conclusion and Future Work

In this dissertation, we have proposed visual learning methods, which are the learning methods that autonomously acquire knowledge for object recognition based on machine learning frameworks. Since visual learning analyzes training examples and constructs recognition models, it is more effective for real-world object recognition tasks than are conventional pattern recognition methods. Specifically, we have proposed two types of visual learning methods based on the SCM algorithm and ensemble learning. The recognition performance of visual learning is highly affected by the machine learning algorithm. Thus, we have proposed an ensemble learning method called NadaBoost, which is suitable for visual learning, by improving AdaBoost's algorithm with respect to its robustness to hard examples. We then applied NadaBoost's algorithm to the proposed visual learning method and verified its effectiveness through several experiments using real-world image data sets.

In Chapter 3, we proposed an effective boosting algorithm, NadaBoost, to resolve the overfitting problem of AdaBoost. We set the threshold $HighWeight$ in each round so that the weights of hard examples which have higher weights than $HighWeight$ are reduced and the theoretical upper bound of the classification error of NadaBoost is lower than that of AdaBoost. Through several experiments, we confirmed that NadaBoost is more robust to noisy data sets which include many hard examples and performs better than NadaBoost even if the data set has few hard examples.

The way of determining the best $HighWeight$, however, is very costly with respect to time complexity. Thus, we introduced the beam search to reduce the computational cost. However, it is still time-consuming compared with AdaBoost. Although the time complexity can be reduced if we use a fixed ad hoc value which is determined empirically for $HighWeight$, the optimal value of $HighWeight$ will be dependent on classification tasks. Thus, we should try to find an efficient approach to reduce the time complexity.

In addition, we have not given theoretical evidence in terms of the generalization error of NadaBoost. Freund and Schapire [11] have shown the method to bound the test error of AdaBoost in terms of its training error, the number of training examples, the VC-dimension of the weak hypothesis space, and the number of rounds. Schapire et al. [32] have introduced the idea of margins of training examples and proven that larger margins on the training examples result in lower test error. We should investigate the theoretical test error of NadaBoost from these viewpoints.

In Chapter 4, we proposed a visual learning method based on the SCM algorithm and introduced the effective feature selection method. Additionally, we have introduced the beam search to reduce the computational complexity. We have verified by several experiments that the feature selection shortens the learning time considerably without the degradation of the classification accuracy. We confirmed from the result that feature selection enables us to find only appropriate attributes which make the hypothesis accurate. In addition, we have introduced the MDL-based cost function to improve the performance of our method by eliminating useless features. By using the cost function, the search for features is terminated at the proper point, so that the hypothesis consists only of useful features. Consequently, the learning time is shortened and the classification accuracy is not changed.

Though our method is highly efficient, we did not find the optimal parameter settings for the maximum number of filters D and *beam_width* for the tradeoff between the classification accuracy and the learning time. While it is very difficult to find the optimal parameter settings because the optimal settings generally depend on the given data set, we should try to find some criterion for the tradeoff.

Since the images in the SAR data set are rather noisy, we confirmed from the experimental results that our method works well for noisy data sets. However, these results are not sufficient to prove that our method is robust to noisy data. Thus, we should carry out more experiments using noisy data sets and make our method more stable.

In Chapter 5, we proposed an effective visual learning model. We modularized the recognition process to recognize complex objects efficiently. Since hard examples have an extreme effect on the recognition performance, we developed an efficient ensemble learning model to reduce the influence of hard examples by introducing collaboration among multiple classifiers. We divided hard examples into two types: true hard examples and feature-dependent hard examples. The true hard examples should be eliminated from training sets because they often cause overfitting, while the feature-dependent hard examples can be utilized to train classifiers. Thus, we proposed an efficient interaction algorithm to eliminate the true hard examples and utilize the feature-dependent hard example.

In the object recognition experiments, we verified the effectiveness of our visual learning model by comparing our recognition performance with that of other object recognition methods. Then, we examined the robustness of our model to hard examples.

The recognition performance, however, is still inadequate to discriminate a wide variety of objects correctly in the real-world scene. Although the recognition performance is highly dependent on the combination of the classifiers, finding the optimal combination is an open problem. Thus, we should develop an efficient method to determine the optimal combination in order further to improve the recognition performance.

In addition, our method has the problem of high computational complexity. The problem of the learning time can be solved by introducing parallel processing into the collaborative ensemble learning. However, we should try to find a more appropriate method of reducing the computational complexity by developing a more efficient learning framework.

Through this dissertation, we discuss the effectiveness of the introduction of learning frameworks into object recognition tasks. This method is effective because human beings acquire their excellent recognition ability through learning. Although the performance of the proposed recognition methods is still inadequate, we attempt to make the recognition ability of visual learning comparable to that of human beings by analyzing and modeling human visual systems.

Acknowledgements

I would like to give my sincere thanks to my advisor Professor Kuniaki Uehara. I have been able to complete this doctoral dissertation because of his guidance and open mind.

I would also like to express my grateful thanks to Professor Yasuo Ariki, Professor Takenao Ohkawa and Professor Hisashi Tamaki, who spent their precious time reviewing this doctoral dissertation.

I am grateful to Associate Professor Yoshiaki Yasumura and Assistant Professor Kazuhiro Seki. They gave the opportunity and knowledge to enhance my study.

I am also grateful to Office Administrator Yoko Maruyama. She often took care of my paperwork and loosened up the atmosphere of our laboratory with her vitality.

I would like to thank all the students in my laboratory. We shared many pleasures and pains in my laboratory life. I wish them continued success and happiness in the future.

Finally, I would like to express my deepest thanks to my family and my friends for their infinite encouragement and kindness to me.

Bibliography

[1] S. Belongie, J. Malik, and J. Puzicha. Matching shapes. In *Proc. of the 8th IEEE International Conference on Computer Vision*, pages 454–463, 2001.

[2] S. T. Bow. *Pattern Recognition and Image Preprocessing.* Signal Processing and Communications Series. Marcel Dekker Inc., 2002.

[3] Intel Corp. Intel image processing library: Reference manual, 2000.

[4] Intel Corp. Open source computer vision library: Reference manual, 2001.

[5] J. Czyz, J. Kittler, and L. Vandendorpe. Multiple classifier combination for face-based identity verification. *Pattern Recognition*, 37:1459–1659, 2004.

[6] T. G. Dietterich. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, 10(7):1895–1923, 1998.

[7] T. G. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning*, 32(1):1–22, 1999.

[8] T. G. Dietterich. Ensemble methods in machine learning. In *Proc. of 1st International Workshop on Multiple Classifier Systems*, pages 1–15, 2000.

[9] Carlos Domingo and Osamu Watanabe. Madaboost: A modification of adaboost. In *Proc. of 13th Annual Conference on Computer Learning Theory*, pages 180–189, 2000.

[10] Center for Imaging Science. MSTAR SAR database, 1997.

[11] Y. Freund and R.E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[12] K. Grauman and T. Darrell. Efficient image matching with distributions of local invariant features. In *Proc. of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 627–634, 2005.

[13] T. K. Ho, J. J. Hull, and S. N. Srihari. Decision combination in multiple classifier systems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16:66–75, 1994.

[14] K. Krawiec. Pairwise comparison of hypotheses in evolutionary learning. *Proc. of the 18th International Conference on Machine Learning*, pages 266–273, 2001.

[15] K. Krawiec and B. Bhanu. Visual learning by evolutionary feature synthesis. *Proc. of the 20th International Conference on Machine Learning*, pages 376–383, 2003.

[16] B. Leibe and B. Schiele. Analyzing appearance and contour based methods for object categorization. In *Proc. of International Conference on Computer Vision and Pattern Recognition*, pages 409–415, 2003.

[17] H. Li and N. Abe. Generalizing case frames using a thesaurus and the MDL principle. *Computational Linguistics*, 3:217–244, 1998.

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

[19] M. Marchand and J. Shawe-Taylor. Learning with the set covering machine. *Proc. of the 18th International Conference on Machine Learning*, pages 345–352, 2001.

[20] M. Marchand and J. Shawe-Taylor. The set covering machine. *Journal of Machine Learning Research*, 3:723–746, 2002.

[21] R. Marée, P. Geurts, J. Piater, and L. Wehenkel. Random subwindows for robust image classification. In *Proc. of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 34–40, 2005.

[22] D. Marr. *Vision*. W.H.Freeman and Company, 1982.

[23] L. Mason, J. Baxter, P. Bartlett, and M. Frean. Boosting algorithms as gradient descent. *Advances in Neural Information Processing Systems*, 12:512–518, 2000.

[24] H. Nomiya and K. Uehara. Feature construction and feature integration in visual learning. In *Proc. of ECML2005 Workshop on Sub-symbolic Paradigm for Learning in Structured Domains*, pages 86–95, 2005.

[25] H. Nomiya and K. Uehara. Multistrategical image classification for image data mining. In *Proc. of International Workshop on Multimedia Data Mining*, pages 22–30, 2007.

[26] J. Peng and B. Bhanu. Closed-loop object recognition using reinforcement learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20:139–154, 1998.

[27] J. R. Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann Publishers, 1993.

[28] J.R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.

[29] J.R. Quinlan. Bagging, boosting and c4.5. In *Proc. of 13th National Conference on Artificial Intelligence*, pages 725–730, 1996.

[30] G. Ratsch, T. Onoda, and K.R. Muller. Soft margins for adaboost. *Machine Learning*, 42(3):287–320, 2000.

[31] R.E. Schapire. Theoretical views of boosting. In *Proc. of 4th EuroCOLT'99*, pages 1–10, 1999.

[32] R.E. Schapire, Y. Freund, P. Bartlett, and W.S. Lee. A new explanation for the effectiveness of voting methods. In *Proc. of 14th International Conference on Machine Learning*, pages 322–330, 1997.

[33] R.E. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. In *Proc. of 11th Annual Conference on Computational Learning Theory*, pages 80–91, 1998.

[34] B. Schiele and J. L. Crowley. Recognition without correspondence using multidimensional receptive field histograms. *International Journal of Computer Vision*, 36(1):31–50, 2000.

[35] M. J. Swain and D. H. Ballard. Color indexing. *International Journal of Computer Vision*, 7(1):11–32, 1991.

[36] Z. Tu. Probabilistic boosting-tree: Learning discriminative models for classification, recognition, and clustering. In *Proc. of the 10th IEEE International Conference on Computer Vision*, pages 1589–1596, 2005.

[37] M. A. Turk and A. P. Pentland. Face recognition using eigenfaces. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, pages 586–591, 1991.

[38] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, pages 511–518, 2001.

Publication List

Book

1. Hiroki Nomiya and Kuniaki Uehara,
   "Content-based Image Classification via Ensemble Visual Learning,"
   ed. Vedran Kordic, Data Mining; From Data to Knowledge, I-Tech Education and Publishing (provisional acceptance).

Journals

1. Masayuki Nakamura, Hiroki Nomiya and Kuniaki Uehara,
   "Improvement of Boosting Algorithm by Modifying the Weighting Rule,"
   Annals of Mathematics and Artificial Intelligence, Vol.41, No.1, pp.95-109 (2004).

2. 中村雅之，野宮浩揮，上原邦昭,
   "重み更新規則の修正による Boosting アルゴリズムの改善について,"
   情報処理学会論文誌, Vol.45, No.3, pp.1001-1013 (2004).

3. 野宮浩揮，上原邦昭,
   "相補的な視覚的学習による複数の認識手法の統合,"
   電子情報通信学会論文誌, Vol.J90-D, No.11, pp.3043-3054 (2007).

4. 中村哲也，瀧敬士，野宮浩揮，上原邦昭,
   "AMSS: 多次元時系列の効率的な類似度測定手法,"
   電子情報通信学会論文誌, Vol.J91-D, No.11 (採録決定).

5. Hiroki Nomiya and Kuniaki Uehara,
   "Visual Learning Using Collaboration of Multiple Classifiers,"
   Journal of Computer Vision and Image Understanding (under review).

Proceedings

1. Hiroki Nomiya and Kuniaki Uehara,
   "Visual Learning by Set Covering Machine with Efficient Feature Selection,"

Proc. of 16th European Conference on Artificial Intelligence, pp.525-529 (2004).

2. Hiroki Nomiya and Kuniaki Uehara,
“Feature Construction and Feature Integration in Visual Learning,”
Proc. of ECML-2005 Workshop on Sub-symbolic Paradigm for Learning in Structured Domains, pp.86-95 (2005).

3. Hiroki Nomiya and Kuniaki Uehara,
“Multistrategical Image Classification for Image Data Mining,”
Proc. of International Workshop on Multimedia Data Mining, pp.22-30 (2007).

4. Hiroki Nomiya and Kuniaki Uehara,
“Multistrategical Approach in Visual Learning,”
Proc. of the 8th Asian Conference on Computer Vision, pp.502-511 (2007).

Oral Presentations

1. （査読なし）野宮浩揮, 上原邦昭,
“アンサンブル学習の適用による Linear Discriminant Analysis アルゴリズムの改善,”
情報処理学会 数理モデル化と問題解決研究報告, pp.17-20 (2003).

2. （査読なし）野宮浩揮, 上原邦昭,
“重み付きマッチング法を用いた特徴構成法による視覚的学習,”
電子情報通信学会 2005 年総合大会講演論文集, D-12-96 (CD-ROM) (2005).

3. （査読なし）野宮浩揮, 上原邦昭,
“木構造に基づく特徴構成法による視覚的学習,”
第 19 回人工知能学会全国大会論文集, 2E2-01 (CD-ROM) (2005).

4. （査読なし）野宮浩揮, 上原邦昭,
“形状ベースと領域ベースの認識手法の統合による視覚的学習,”
第 9 回画像の認識・理解シンポジウムダイジェスト, pg.464 (2006).

5. （査読あり）野宮浩揮, 上原邦昭,
“アンサンブルアプローチに基づく多戦略視覚的学習,”
第 10 回画像の認識・理解シンポジウムダイジェスト, pp.271-278 (2007).

6. （査読なし）野宮浩揮, 上原邦昭,
“相補的な学習機構に基づく協調型アンサンブル学習法,”
第 22 回人工知能学会全国大会論文集, 2J2-4 (CD-ROM) (2008).