ABSTRACT

SVM-based incremental learning, which can make a user-relevant recognition system quickly adapt to specific users' preference without losing its general performance, is an elegant solution for user adaptation problems in on-line graphics recognition system. Two learning strategies (repetitive learning and incremental learning), two incremental learning algorithms (Syed et al.'s and Xiao et al.'s), and two classifier structures (one-against-one and one-against-all) are compared under the multi-class classification environment. Theoretical analysis and experimental results both show: (1) incremental learning can adapt the classifier to new obtained samples much faster than repetitive learning without losing any precision; (2) the SVM-based incremental learning algorithm of Syed et al.'s is superior to that of Xiao et al.'s; (3) one-against-one structure is superior to one-against-all structure for a multi-class incremental learning environment.

1. INTRODUCTION

User adaptation is a classical and important problem in user interface study. Many pattern recognition problems deal with users' specialization, for users' handwritings, drawing styles, and accents are different. The purpose of user adaptation is to recognize the users' intention. The intentions of different users are various and even inconsistent. Take on-line graphics recognition [1] as an example. Different users have different drawing habits/preferences. The user1 want to input a quadrangle and the user2 is just to draw a triangle while their sketches are similar, as in Figure 1. Obviously, in order to identify the users’ intents, it is necessary to build the user adaptation system.
experience much better (with more smooth interaction). For a multi-class classification problem, one-against-all structure is not a wise solution for incremental learning, for it is much more time consuming than one-against-one structure in our research.

In this paper, we introduced SVM-based incremental learning for user adaptation in a multi-class classification environment for on-line graphics recognition. The repetitive learning strategy and other two incremental learning algorithms are also compared. In addition we compare one-against-one structured multi-class classifier with one-against-all one. Finally, based on the theoretical analysis and experimental results, we draw the conclusion that SVM-based incremental learning can adapt to multi-user in on-line graphics recognition system.

The paper is organized as following. Section 2 presents related works. Section 3 presents a brief introduction of SVM and two SVM-based incremental learning algorithms. Our solution for user adaptation, which is based on a multi-class SVM classifier with incremental learning ability, is presented in Section 4. Experimentation and performance evaluation are given in Section 5. Finally, section 6 draws conclusions.

2. RELATED WORKS

Support Vector Machine (SVM) is a new and promising pattern recognition technique developed by Vapnik and his research group [3]. SVM is based on the theory of Structural Risk Minimization, and can gain good performance in little samples without over fitting.

As an ability to handle the issues of scalability and expandability of learning systems, incremental learning has attracted much attention in recent years. It has two advantages: computation efficiency and storage economy. Most incremental learning Algorithms is based on decision-trees and neural networks[4][5][6][7]. These Algorithms are short of expected risk controlling mechanism over the whole sample set, and they could not discard samples optimally. Syed et al.[8] retrain classifier with all SV and newly input samples. Xiao et al.[9] proposed a different incremental learning approach for SVM based on the boosting idea. We will compare these two algorithms and their computation complexities in section 4, and then give evaluation in section 5.

Over the past years, progress has been made to construct multi-class classifier based on the SVM theory. Except for the two classical structures of one-against-all and one-against-one, Platt et al.[10] adopted the Decision Directed Acyclic Graph (DDAG) to combine many binary (two-class) classifiers into a multi-class classifier. Weston et al.[11] have proposed an extension to solve multi-class classification problems in one step. Generally speaking, these approaches are either too complicated or time consuming. One-against-all structure is the easiest to be implement, and most current multi-class utilities, such as SVMTorch[12] and SVMLight[13], are based on one-against-all structure. However, this structure is not suitable for incremental learning purpose. We will analyze this problem in section 4 and section 5.

3. THE SVM-BASED INCREMENTAL LEARNING ALGORITHM

3.1 Brief Introduction of SVM

The main idea of Support Vector Machine (SVM)[3] is to construct a nonlinear kernel function to map the data from the input space into a possibly high-dimensional feature space and then generalize the optimal hyper-plane with maximum margin between the two classes. Hence, it is basically used for binary (positive or negative) classification.

The equation of the optimal hyper-plane separating the two classes can be expressed by \( w \cdot x + b = 0 \), where \( w \) denotes the normal of the hyper-plane, \( b \) denotes the offset. The training data samples are represented by \( (x_1, y_1), \ldots, (x_n, y_n) \), where \( x_i \in \mathbb{R}^d \) denotes a vector (data sample) in a \( d \)-dimensional space and \( y_i \in \{-1, +1\} \) represents the class to which \( x_i \) belongs. From the Kuhn-Tucker condition, we know that \( w \) can be expressed by a linear combination of the training samples, as is illustrated in Eq. 1.

\[
\alpha = \sum_{i=1}^{n} \alpha_i y_i x_i, \quad 0 < \alpha \leq C
\]  

(1)

The objective of training is to obtain each sample’s \( \alpha \) value. Classification is done based on the test sample’s distance to the hyper-plane. Eq. 2 illustrates the decision function which used to classify the test sample \( x \).

\[
f(x) = \text{sgn}\left( \sum_{i=1}^{n} \alpha_i y_i (x \cdot x_i) + b \right)
\]  

(2)

If \( f(x) = +1 \), \( x \) is classified as positive; if \( f(x) = -1 \), \( x \) is classified as negative.

In most cases, the sample space is not linearly separable. A mapping \( \Phi \) is usually used to non-linearly transform the input samples into a high dimensional feature space so as to make these samples linearly separable. Thereby the format of the decision function has been changed to:

\[
f(x) = \text{sgn}\left( \sum_{i=1}^{n} \alpha_i (\Phi(x) \cdot \Phi(x_i)) + b \right)
\]  

(3)

where \( K(x, x_i) = \Phi(x) \cdot \Phi(x_i) \) is known as the kernel function. Usually, only a small portion of samples have non-zero \( \alpha \) coefficients, whose corresponding \( x_i \) (a.k.a., support vectors) and \( y_i \) fully define the decision function. Therefore, the SV set can fully describe the classification characteristics of the entire training set. Because the training process of SVM involves solving a quadratic programming problem, the computational complexity of
the training process is much higher than a linear complexity (wrt the number of training samples). Hence, if we train the SVM on the SV set instead of the whole training set, the training time can be reduced greatly without much loss of the classification precision. This is the main idea of our incremental learning algorithm.

In the training process (denoted as $\text{Train}(\mu)$), the training set $\mu$ is divided into two sub-sets: the SV set ($\mu_{sv}$) and the non-SV set ($\mu_{non-sv}$). If most SVs are not on the classification boundary, the computational complexity of the training process can be specified in Eq. 4.

$$O(\|\mu_{sv}\| + |\mathcal{X}| \times \|\mu_{sv}\| + d |\mathcal{X}| \times \|\mu_{sv}\|) \tag{4}$$

Generally speaking, $\|\mu_{sv}\| << |\mathcal{X}|$ and they are relevant variants. Therefore, the training complexity can be specified in Eq. 5.

$$O(\|\mu_{sv}\|) \tag{5}$$

The classification process (denoted as $\text{Classify}(\nu)$) of an SVM classifier can be specified as dividing the test data set $\nu$ into the following two sub-sets.

$$\nu_{sv} = \{ (x_i, y_i) \mid (x_i, y_i) \in \nu, y_i f(x_i) \geq 0 \}$$

$$\nu_{error} = \{ (x_i, y_i) \mid (x_i, y_i) \in \nu, y_i f(x_i) < 0 \}$$

The classification precision $p$ is defined as

$$p = \frac{|\nu_{sv}|}{|\nu|} \tag{6}$$

The classification complexity is

$$O(\|\nu_{sv}\|) \tag{7}$$

For a SVM classifier $\Gamma$, denoting its initial training set as $\text{IS}$, and its incremental training set as $\text{INS}$, we define two learning strategy: one is repetitive learning, whose training process can be specified as $\text{Train}(\text{IS}, \text{INS})$; the other is incremental learning, whose training process can be specified as $\text{Train}(\text{Sub}(\text{IS}), \text{INS})$, where $\text{Sub}(\text{IS}) \subset \text{IS}$.

A basic assumption for incremental learning is that the incremental training set does not change classification boundary very much. Otherwise, repetitive training may be a better choice. If this pre-condition is met, we obtain

$$\|\text{IS} \cup \text{INS}\| \approx \|\text{IS}_{sv}\| + \|\text{INS}\| \tag{8}$$

And in most cases, $|\text{INS}| << |\text{IS}|$. Therefore, the computational complexity of the repetitive learning is:

$$O(\|\text{IS} \cup \text{INS}\| \times \|\text{IS}_{sv}\|) = O(\|\text{IS}_{sv}\|) \tag{9}$$

In simplification, we focus on binary classification problem in our discussion on incremental learning, which can easily be extended to multi-class classification problem. Four extra sample sets are used in our further discussion. They are working set ($\text{WS}$), backup set ($\text{BS}$), new set ($\text{NS}$), and discarded set ($\text{DS}$), respectively.

### 3.2. Incremental Learning Algorithms for SVM

Xiao et al.[9] proposed two major issues in the incremental SVM learning algorithms:

1. How to utilize the historical training results to re-train the classifier more quickly?
2. How to discard samples optimally to reduce storage space without losing much precision?

Syed et al.[8] proposed an incremental SVM learning algorithm, which uses only the historical SV samples and the incremental training samples in re-training. All non-SV samples are discarded after previous training. The algorithm can be illustrated in algorithm 1 as follows.

**Table 1. Syed et al.’s algorithm**

<table>
<thead>
<tr>
<th>Algorithm 1: Syed et al.’s algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1. $\Gamma$=Train(\text{IS}), WS=\text{IS}_{sv}$;</td>
</tr>
<tr>
<td>Step 2. WS=WS\cup\text{INS};</td>
</tr>
<tr>
<td>Step 3. $\Gamma$=Train(WS), WS=WS_{sv};</td>
</tr>
</tbody>
</table>

Steps 2 and 3 form the incremental training/learning process. The computational complexity of Algorithm 1 is

$$O(|\text{INS}| \times |\text{IS}_{sv}|) \tag{10}$$

Xiao et al.[9] proposed a different incremental SVM learning algorithm based on the boosting idea. This algorithm is listed as the following Table 2. Such redundant samples, which are far from the hyper-plane, must be discarded from BS according to a discarding rate $r$ ($r$ is usually set at 0.5–0.9, depending on different situations) to reduce the re-classification time and storage space. Denote the discarded sample set as $\text{DS}$. We define $\text{DS}=\text{discard}(\text{BS})$. The discard process is executed only once after the first time re-classification. The algorithm is illustrated as follows.

**Table 2. Xiao et al.’s algorithm**

<table>
<thead>
<tr>
<th>Algorithm 2: Xiao et al.’s algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1. $\Gamma$=Train(\text{IS}), WS=\text{IS}<em>{sv}, BS=\text{IS}</em>{non-sv}$;</td>
</tr>
<tr>
<td>Step 2. $\text{NS}=\text{INS}, \text{firsttime}=\text{TRUE}$;</td>
</tr>
<tr>
<td>Step 3. $\text{Classify}(\text{NS})$ by $\Gamma$, WS=\text{NS}<em>{sv}\cup\text{INS}, BS=\text{NS}</em>{sv}\cup\text{BS}$;</td>
</tr>
<tr>
<td>Step 4. If $p &gt; \beta$ then Stop;</td>
</tr>
<tr>
<td>Step 5. If firsttime then $\text{DS}=\text{discard}(\text{BS})$, BS=BS-DS, firsttime=FALSE;</td>
</tr>
<tr>
<td>Step 6. $\Gamma$=Train(WS), WS=WS_{sv}, BS=BS, WS=WS_{non-sv};</td>
</tr>
</tbody>
</table>

If we use $k$ to denote the iteration times between step 3 to step 7, the computational complexity of algorithm 2 is

$$O((k-1)|\text{IS}_{sv}| + |\text{INS}_{sv}| |\text{IS}_{sv}| + kd(1-r)|\text{IS}_{sv}|) \tag{11}$$

### 4. SVM-BASED INCREMENTAL LEARNING FOR USER ADAPTATION

#### 4.1 Feature Extraction

For simplification, we have the following assumption in on-line graphics recognition system:

1. The sketch is closed-shaped and composed of lines.
2. The sketch is finished in one stroke, and only a basic shape such as ellipse, triangle, quadrangle,
pentagon or hexagon.
3. All polygons with the same vertex number are included in the same shape type. For instance, all squares, rectangles, trapezoids, and parallelograms are regarded as quadrangles.
4. No hooklet in the beginning or ending of one stroke (as Figure 2(c)).
5. No circlet at the corner of the two lines (as Figure 2(d)).

Figure 2. Improper Shapes. (a) a pentagon with a cross, (b) an un-closed triangle, (c) a hooklet at the beginning or ending of the stroke, (d) a circlet at the corner.

However, user may not follow the above rules we have defined, hence we need preprocess the input shape in order to simplify the process of extracting feature.

Obviously, the features we used for recognition is irrelevant with polygon’s size, position, and rotation direction. Therefore, turning function[14] is employed to obtain the feature vector of the polygon representation of the sketchy stroke.

Turning function ΘA(s) measures the angle of the counterclockwise tangent as a function of the arc-length s, staring from a reference point O on a polygon A’s boundary. Thus ΘA(0) is the angle v of the tangent at O from the x-axis, as is showed in Figure 3. ΘA(s) accumulates the turning angles (which is positive if the turning is left-hand and negative otherwise) as s increases.

Our definition of turning function is a little different from the commonly used one[14]. We see that the previous definition is relevant to both the rotation orientation of the polygon and the reference orientation. We scaled the polygon so that its perimeter length is 1. We use the tangent at O as the reference orientation such that ΘA(0)=0 and determine a traversing direction such that all turning angles are positive. Hence, our turning function ΘA'(s) is a monotonous increasing one from [0, 1] into [0, 2π].

Figure 3. A polygon and its turning function

If we use an m-dimensional feature vector, we equally divide the boundary of the polygon into m pieces. Each element of the feature vector is the turning degrees accumulated in a corresponding piece of the polygon, as defined in Eq. 12.

\[ V_i = \begin{cases} 
\phi_i, & \text{if } \phi_i \geq 0 \\
\phi_i + 2\pi, & \text{if } \phi_i < 0 
\end{cases} \quad (0 \leq i \leq m) \]  

(12)

where \( \phi_i = \Theta_i' \left( \frac{i \mod m}{m} \right) - \Theta_i' \left( \frac{i-1}{m} \right) \) and m is the dimension.

4.2 SVM-based Incremental Learning for User Adaptation

In the on-line graphics recognition system, the sketches are inputted and recognized one by one. Obviously, it is inefficient for the system begins to learn users’ habit as soon as one shape is inputted. However, the precision will be poor if the system begins to learn after gathered a great deal of samples. Hence, we limit of the scale of samples. The system will begin to learn users’ preferences once scale limits are reached, and continue till a stable system status was obtained.

Now there are three strategies to solve the problem of learning users’ preferences:

Method 1. Repetitive learning, i.e. retrain all historical samples and new ones.
Method 2. Incremental learning Algorithm 1.
Method 3. Incremental learning Algorithm 2.

It is difficult and useless to discuss the complexity of the three strategies because the sample is small in the beginning of the system. If the new data set (INS) is much smaller than the historical training data set (IS), i.e. \( |INS| \ll |IS| \), then:

\[ \frac{|INS \cup IS_i|}{|IS_i|} \approx \frac{|IS|}{|INS|} \]  

(13)

\[ \frac{(k-1)|IS_1 + INS_j|}{|IS_j|} \approx kd(1-\tau) \cdot (k-1)|IS_i| \]  

(14)

From Eq. 13 and 14, we find that the training time of Algorithm 2 is k-1 times larger than Algorithm 1 where k is an unpredictable coefficient. Since the re-training process of Algorithm 2 only adapts to the mis-classified samples in the incremental training set, correct-classified samples in previous iteration steps may be mis-classified in later iterations. This makes some samples switch between the working set and the backup set repeatedly and thus makes k very large, which results in the much lower speed of Algorithm 2 than that of Algorithm 1. On the other hand, because \( |IS_{SV}| \ll |IS| \), and k is an integer usually vibrating between 2 and 5, Eq. 14 is still less than 1, and Algorithm 2 is still faster than repetitive training.

In the previous discussion we only focused on binary-classification problem. However, real application problems are usually multi-classed. Two major structures, one-against-all and one-against-one, are usually used to build the multi-class classifier with many binary sub-classifiers. For one-against-all structure (m classes and n samples in the training sets), m sub-classifiers are
adaptation and give the answers based experiments. In the one-against-one method, which is based on the max-win scheme[11], each sub-classifier casts one vote for its preferred class, and the final result is determined by the class with the most votes. Compared with traditional one-again-all structure, one-against-one has many advantages, as is shown in Table 3.

In this study, we select the method 2 and one-against-one structure in multi-class classifier to adapt to multi-user in on-line graphics recognition system. The following are the detailed steps:

1. Collecting samples until they are enough to train the classifier.
2. Learning users’ preferences.
   1) Build a classifier for every two classes.
   2) Train or retrain the classifiers with the historical SV and newly collected samples.
3. Recognizing and gathering samples.
   1) Classify with every sub-classifier.
   2) Vote by every sub-classifier and the final result is the class with the most votes.
   3) Gather the evaluation of the user and store the user’s data.
4. Goto Step 2.2 until enough samples are obtained.

5. EXPERIMENTAL RESULTS

Now, we have the following four questions for user adaptation and give the answers based experiments.

**Question 1:** Does there exist conflicts among the shape classification precisions for different uses?

**Question 2:** Can the incremental learning save much of the training time without losing any precision?

**Question 3:** Which one is better, Algorithm 1 or Algorithm 2?

**Question 4:** Which structure of the multi-classes classifier is better, one-against-one or one-against-all?

5.1. Experimental Environments

In total, we collect 1367 sample shapes for user1 and 325 samples for user2 of the five classes. All samples are extracted into a 20-dimensional feature by turning function. We further obtain virtual samples through the following transformation.

1) **Rotation:** Feature vector are relevant to start point selection. By shifting the start point to each of the turning (sampling) points, we obtain 19 addition virtual samples for each sample.

2) **Mirroring:** Reversing the feature vector’s dimension sequence, a virtual sample can be obtained for each (virtual or real) sample.

For detailed information of the feature extraction process and the transformation of the samples, please refer to [1]. For each sample, we now have 40 samples after transformation. Therefore we have 54680 samples for user1 and 13000 samples for user2 altogether. We randomly select 19210 samples of user1 to form a test set \( T_1 \) and use the all samples of user2 to form a test set \( T_2 \). Then we randomly select samples from the remained samples of user1 to form 40 incremental training sample sets. The first 6 incremental training sets have 100, 100, 120, 150, 300, and the later 34 sets have 1000 samples respectively. This is reasonable for an incremental learning process because SVM have the ability to obtain a good performance under a small training set and it is more difficult to improve its precision from the incremental train set after the classifier maintained a relatively stable precision.

In the following experiments, a RBF kernel[3] is used and the implementation of the training process Train() is the same as SVMTorch[12]. All experiments are done on an Intel P4 PC (with a 1.4G Hz CPU and 256MB memory) running Microsoft Windows 2000.

5.2. Rule-based Relevance Feedback

For rule-based approaches, some parameters are available to be adjusted through user feedback. Unfortunately, although user feedback can adapt the system to a new user, the system may lost its universality to other users. With the variation of the threshold, the shape classification precisions for user1 and user2 also change. If the threshold is increased from 0.35 to 0.75, the precision for user 2 decreased, while the precision for user2 increased. This phenomenon means that no matter what kind of feedback strategy is employed, the system cannot raise its performance for both user1 and user2 so that their corresponding precision curves all reaches their peaks (91.2% at the threshold 0.75 for user1 and 92.6% at 0.35 for user2). Consequently, rule based approaches are not powerful enough to reflect very complex relations due to their intrinsic limitations such as scalability and adaptability (only reach 90.8% at 0.65 for both user1 and user2). This is illustrated in Figure 4. Intuitive feedback strategy usually results in Ad Hoc situations, which gained “personalized” performance on the expense of the loss of its universality.

5.3. SVM-based Incremental learning

In experiments, we compare the performance among different incremental algorithms and repetitive learning mentioned in section 4.2 on the two multi-class classifiers for five classes. We train the classifier on the 40 incremental training sets. Then we compare the training time in both multi-class classifiers, average open-test precision for User1 in both multi-class classifiers, average open-test precision for User2 in both multi-class classifiers. The results are shown in Figure 5.

5.4. Performance Evaluation

Base on these experiment results, we can answer the previous presented four questions.
1. The answer to Question 1
   There are no obvious conflicts among precision of different users. Take configuration 2 as an example (for the other two configurations, we have very similar results), the shape classification precisions of user1 and user2 approximately both rise with the incremental steps (although there may be some fluctuation of the precision curve). When the classifier obtained a highest precision of user1, which is 95.92% in our experiment, it also gained a high highest precision of user2, which is 96.56%. The precision is 96.2% at average for both User1 and User2. Consequently, when the classifier further adapts user1’s preference, it does not lose its general performance, as is illustrated in figure 6.

2. The answer to Question 2
   Incremental learning does work. First, training time can be much saved when using incremental learning instead of repetitive training. From Figure 5(a) we can see that repetitive learning is much more time-consuming than other two incremental learning algorithms, and its training time increases sharply when the number of training samples increased. Second, incremental learning can help the system adapt to the new training samples. The precision curves of user1 and user2 both rise under the three configurations. And little precision is lost for incremental learning compared with repetitive learning. Refer to Figure 5(b) and Figure 5(c), the precision curve of configuration 2 and 3 do not vary much from configuration 1 during the whole incremental learning processes.

3. The answer to Question 3
   As we expected, Algorithm 1 is superior to Algorithm 2 in its training speed. From Figure 4.2(a) we can draw a conclusion that Algorithm 2 is much slower than Algorithm 1. Moreover, unlike Algorithm 1, the training time of Algorithm fluctuates violently and unpredictably. This is because the unpredictable coefficient $k$ of Eq. 14, which makes the training time of Algorithm 2 is usually twice of Algorithm 1. As precision is concerned, there is no obvious difference between Algorithm 1 and Algorithm2. And we find that the precision curve of Algorithm 1 is usually a little above algorithm 2. This means the precision of Algorithm 2 is no better than that of Algorithm 1 (Cf. Figure 5(a) and 5(b)).

4. The answer to Question 4
   As we have analyzed, one-against-one structure is more efficacious than one-against-all structure for multi-class classification problems. According to theoretical analysis in our experimental environment, the incremental training time of one-against-all structured incremental classifier should be at least 10 times longer than that of one-against-one structured one. Although the difference is not as large as we have expected, we can still find that one-against-one structured classifier is much faster than one-against-all structured one, as is shown in Figure 7(a).

   On the other hand, the precision curve of the one-against-one structured classifier is always a little above the one-against-all structured one (cf. Figure 7(b)). This is because sometimes the sub-classifiers of the one-against-all structured multi-class classifier will provide conflicting classification results.

6. CONCLUDING REMARKS AND FUTURE WORKS
   User adaptation is a common problem in user interface study and pattern recognition. SVM-incremental learning is a good solution to this problem. Compared with repetitive learning, incremental learning can save much time without any loss of precision. Syed et al.’s algorithm is superior to Xiao et al.’s. We also draw the conclusion that one-against-one structure is more suitable for incremental learning than one-against-all structure.

   Another discover is that the learning ability of the classifier drops obviously after several incremental learning steps, as can be seen from any incremental learning precision curves. This happened because it becomes more and more difficult to get valuable information for the classifier to learn. New learning strategy, which is referred to as Aggressive Learning in this paper, is proposed to solve this problem. Together with incremental learning ability, an SVM-based aggressive classifier will automatically select the most ambiguous samples (which are near the hyper-plane in the SVM theory) for the user to decide, and then employs these samples in incremental training, thus to make the system even more quickly adapt to users by the fewest human interactions. We plan to fulfill this strategy in our future work.

REFERENCES


Table 3. Comparison between one-against-one and on-against-all structures (Supposing there are m classes and n samples)

<table>
<thead>
<tr>
<th>Structure</th>
<th>One-against-one</th>
<th>One-against-all</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training complexity</td>
<td>Low, Averagely $O(n'/m)$</td>
<td>High, Averagely $O(mn')$</td>
</tr>
<tr>
<td>Classify complexity</td>
<td>$O(m(m-1)/2)$</td>
<td>$O(m)$</td>
</tr>
<tr>
<td>VC dimension</td>
<td>Relatively low</td>
<td>Relatively high Inclining to over-fitting</td>
</tr>
<tr>
<td>Incremental learning</td>
<td>Low, Need to retrain m-1 classifiers, with n/m samples each.</td>
<td>High, Need to retrain m classifiers, with n samples each.</td>
</tr>
<tr>
<td>for a new sample</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 4. Rule-based relevance feedback for user adaptation. (a) Recognition precision (for 5 shape classes each and their average value) of user1 is relevant to the change of the threshold (b) The optimal thresholds for user1 and user 2 are different.
Figure 5. Performance evaluation among the three learning methods under different structures. (a) Training time comparison under one-against-one structure (b) Training time comparison under one-against-all structure (c) Shape classification precision comparison under one-against-one structure for User1 (d) Shape classification precision comparison under one-against-all structure for User1 (e) Shape classification precision comparison under one-against-one structure for User2 (f) Shape classification precision comparison under one-against-all structure for User2
Figure 6. There is no obvious conflict between the precisions of user1 and user2. (a) Shape classification precision of Method 2 under one-against-one structure (b) Shape classification precision of Method 2 under one-against-all structure

Figure 7. Training time and shape classification precision comparison between one-against-one and one-against-all structures. (a) Training time comparison (b) Shape classification precision comparison