Adaptive Sketchy Shape Recognition Based on Incremental Learning

Zhengxing Sun, Shuang Liang, Lisha Zhang
State Key Lab for Novel Software Technology, Nanjing University, Nanjing, 210093, PR China
szx@nju.edu.cn

Abstract

Adaptive sketchy shape recognition is a critical problem in sketch-based user interface. In this paper, an SVM-based incremental learning algorithm is made to solve this problem. Our algorithm utilizes only the support vectors instead of all the historical samples, and selects some important samples from all newly added samples as training data. The importance of a sample is measured according to its distance to the hyper-plane of the SVM classifier. Experimentations and evaluations of our algorithm are presented and show the effectiveness of this algorithm in reducing both the training time and the required storage space for the training dataset to a large extent with very little loss of precision.

Keywords: Sketch-based User Interface, Adaptive Sketch Recognition, Incremental Learning.

1. Introduction

In sketch-based user interaction, the goal is to allow users to naturally express their ideas with freeform drawing. However, sketching is usually informal, inconsistent and ambiguous. Freehand-drawing for the same shape may be quite different from user to user and even from time to time by the same user. A system that is customized for a particular user even may not work for another different user. Hence, it is not surprising that the poor efficiency of recognition engines, even in the latest experiments[1][2], is always frustrating, especially for the newly added users and composite shapes, though numerous researches have been working on this subject for many years and have produced a wide variety of techniques.

In our previous researches, two kernel methods have been developed for our online sketch recognition system to support users' creativities, named Magic-Sketch[1][3]. The rule-based sketchy shape recognition[3] can fits and rectifies gradually the shapes to a regular one based on users' feedback of partial and overall structural similarity of shapes. It can adjust the parameters and thresholds for the systems such that they can yield acceptable results for new users. The SRG (Spatial Relation Graph)-based composite shape recognition approach[4] can be suitable for the shapes with different complexities in our experiments. However, as the graph isomorphism is originally an NP-complete problem, matching time of SRG is very high for real-time interaction in some cases. In fact, making machines learn to understand users' drawn sketches requires sophisticated and highly adapted algorithms of sketch recognition. Benefiting from advances in sketchy shape recognition system could not be expected before the problem of user adaptation is well solved. Therefore, more complex, statistical approaches such as Neural Network and Support Vector Machine are required.

Human cognition is usually performed incrementally and iteratively and to adapt to a particular user's interaction styles means to re-train the recognizer/classifier with training samples obtained for this user. Since the newly introduced sample set is usually much smaller than the previously accumulated one, retraining with all these samples is time-consuming and not economical. In this case, incremental learning is preferable. It mainly involves the newly obtained samples in retraining and makes the recognizer learn fast by avoiding full usage of previous samples. The Support Vector Machine (SVM)[5] is a new but promising technique to these pattern recognition problems. Although SVM has many excellent aspects, which are suitable for incremental learning, unfortunately, classical SVM learning algorithm does not support incremental learning. In order to adapt to new samples, the traditional SVM has to discard all the previous training results and re-train the new classifier on the whole data set. Hence, this method is not economic in time and space. Syed et al[6] and Xiao et al[7] proposed different incremental learning models based on SVM, but both of them have shortcomings in performance.

In this paper, we present an SVM-based incremental learning algorithm and applied it to the problem of adaptive sketchy shape recognition. By extending two available algorithms (Syed et al[6] and Xiao et al[7]), we develop a new incremental learning algorithm. It uses the support vectors instead of all historical samples and selects some important samples from the incremental samples as the training data in the incremental training steps. As a result, both the training time and the required storage space are saved. This algorithm forms the basis of the multi-class classifier implemented in our on-line sketchy graphics recognition system[3][8][9]. Our experimental results show effectiveness and efficiency of this algorithm.

2. Modified Incremental Learning based on SVM Algorithm

Support Vector Machines (SVMs) have been proved

efficient and suitable for learning with large and high dimension data sets. Furthermore, investigations[6][7] have shown that the incrementally trained SVMs are better than their non-incrementally trained equivalents. This rests mainly with the fact that the generalization property of an SVM does not depend on all the training data but only a subset, which is referred to as Support Vectors (SVs). In other words, the number of SVs typically is very small compared with the number of all training samples. In fact, the SVs are those that lie the closest to the hyper-plane[7]. Nevertheless, there is a problem of how to distinguish them from the rest in the sample sets before training. A solution is to minimize the collected data sets (S_{CL}) by making use of the historic result of training, so that the samples are close enough to the hyper-plane as SV sets (S_{SV}). That is, $Min(S_{CL} | S_{SV} \subset S_{CL})$. Additionally, the training process of SVMs itself may be very time consuming, especially when dealing with noisy data, and the samples used in incrementally training of SVM-classifier must be labeled artificially. This is tedious and burdensome, even an interruption of the user's thinking process.

Syed et al[6] proposed an incremental learning model based on SVM, which is to preserve only the support vectors (SV) at each incremental step and add them to the training set for the next step. However, this algorithm cannot maintain a stable performance in a general sense. Xiao et al[7] proposed a new SVM incremental learning method based on the boosting idea. Their approach employs an iterative classification-training process to retrain the classifier. The superiority of this algorithm is that its ultimate close-test precision is guaranteed before the process stops. However, there is no theoretical analysis that guarantees this algorithm can terminate and may be a vibration of samples between the correctly classified sample set and mis-classified sample set.

We utilize the respective advantages of these two algorithms mentioned above and propose our algorithm. The iteration mechanism of Syed et al's algorithm is introduced to utilize historical training samples, where the SVs of the historical samples are used to replace all historical samples in the re-training process. In the iteration, the previously trained classifiers evaluate all newly collected samples (denoted as *INS*) and select the most important instances (denoted as *NS*,) by evaluating all newly added samples using the process *Evaluate()*. Re-training is performed on the chosen samples and the historical SVs. Although Xiao et al's algorithm makes the best use of the previously trained classifiers and selects some important samples from those newly added samples in the re-training process, it fails to notice that the important samples are those that are close to the hyper-plane. In addition, their algorithm is instable. These limitations do not exist in our algorithm.

Accordingly, we propose an SVM-based incremental learning algorithm, which inserts a questioning process in the training process of SVM incremental learning. The training would only preserve the SVs at each incremental step and then add them instead of all historic data to the training sets collected in the questioning step. In other words, SVM classifier analyzes users' incremental unlabeled samples and picks out the most important instances for the user to endorse as training samples. Denoting the training process of SVM as *Train(SV)*, the process of evaluating all newly added samples as *Evaluate*(), the initial training samples as *IS*, the incremental training samples as *INS*, the temporary training samples as *NS* and working data as *WS*, the process of SVM incremental active learning can be briefly described as:

(1). Γ***=Train(IS), WS=IS***$_{SV}$. The random training samples are collected and their correct classes are specified until they are sufficient for training the classifier initially. A sub-classifier for every two classes is built if it does not exist, and is then trained or retrained with the historical SVs and newly added training samples to learn users' preferences of sketching strokes.

(2). Γ***=Train(WS), WS=WS***$_{SV}$. The features of the samples (In sketch recognition, the features are the input strokes described by the modified turning function[10], which will be introduced in 3.2), and are classified with every sub-classifier. The outputs are the classes with the most votes from every sub-classifier. The newly added samples are selected if they are close to the hyper-plane.

(3) .***NS=Evaluate(INS), WS=WS***$\cup$***NS***. The newly added samples are evaluated. The evaluation is stored and gathered as the incremental training samples.

(4). Repeat **(**2) and (3) until the results of SVM classifiers are satisfactory or enough training samples are obtained.

The key in our algorithm is how to select those important samples in the *Evaluate()* process. People usually consider mis-classified samples are the most valuable to SVM classifiers because they think the recognition precision of SVM classifiers can be greatly improved if the classifiers can recognize these mis-classified samples. For example, Xiao et al's algorithm[7] uses the mis-classified samples in every learning step. In fact, some mis-classified samples may be important if they are close to the classification hyper-plane but others are usually useless for retraining the SVM classifiers. This is because SVM only selects those nodes that are close to the classification hyper-plane as Support Vectors. Therefore, important samples are not those mis-classified samples that are far away from the hyper-plane. Moreover, it is not easy to collect these mis-classified samples because if so, the user has to confirm the correct class of every sample. This is not a feasible user scenario. Those new samples that can be correctly recognized and are also close to the hyper-plane are actually important because those samples record the characteristics of the incremental dataset.

Hence, in our incremental learning algorithm, we only select those samples that are close to the classification hyper-plane, as well as the previous SVs, for re-training.

In our evaluating process, the interrogative samples are determined by estimating their distance from the hyper-plane. Given a training set of *n* samples:

$$F=\{(x_1,y_1),\cdots,(x_n,y_n)\},\ y_i\in\{+1,-1\},$$

the original data space can be mapped to a higher dimensional feature space *f* via a Mercer Kernel operator[11]: $K(x,y)=\Phi(x)\cdot\Phi(y)$, where $\Phi:F\to f$, so as to make these samples linearly separable in the higher dimension feature space. We build the hyper-plane that separates the training data by maximization of the margin in the higher dimension feature space *f* and the hyper-plane can then be expressed as:

$$\sum_{x_i\in SV}\alpha_i y_i K(x,x_i)+b=0,\ 0<\alpha_i\le C \qquad (1)$$

That is, $w_0\Phi(x)+b=0,$

where, $w_0=\sum_{x_i\in SV} y_i\alpha_i\Phi(x_i)$,

Coefficients α_i can be obtained from solving the following optimization problem:

Minimize:

$$w(\alpha)=\sum_{i=1}^{l}\alpha_i-\frac{1}{2}\sum_{i,j=1}^{l}\alpha_i\alpha_j y_i y_j\Phi(x_i)\cdot\Phi(x_j), \qquad (2)$$

subject to constraints:

$$\sum_{i=1}^{l}\alpha_i y_i=0,\ \alpha_i\ge 0,\ i=1,2,...,l. \qquad (3)$$

The distance between the sample *x* and the hyper-plane can be defined as:

$$d(x,w)=\frac{|w_0\Phi(x)+b|}{\|w_0\|} \qquad (4)$$

Since $\|w_0\|=1$,

$$d(x,w)=|w_0\cdot\Phi(x)+b|=|\sum_{x_i\in SV}\alpha_i y_i K(x,x_i)+b| \qquad (5)$$

Intuitively, there is a conflict between the precision and the speed if we use a constant distance threshold to choose the important samples. Hence, we consider using a dynamic threshold in our proposed algorithm. In the beginning steps of incremental learning, due to the small size of the training dataset, the time to re-train SVM classifiers is relatively short and the recognition precision of SVM classifiers is also relatively low. In this case, we can use a larger threshold value in the evaluation process such that the precision can be a little bit higher while the time is still acceptable. As the number of training data increases in later incremental learning steps, the cost (in both time and space) of training and the recognition precision will also increase. In this case, the evaluation process should judge the importance of every newly added sample more carefully with a smaller threshold in order to avoid substantial increase of the number samples necessary for re-training of the SVM classifiers. In another aspect, if we want to adapt to a new user, we can use a large threshold value to adapt him quickly. Using this dynamic threshold method, we can obtain higher precision with shorter training time.

The computing complexity of the evaluation process is:

$$O(\|INS\|\times\|IS_{sv}\|)$$

The complexity of the evaluation process is much less than the training complexity. So, the computational complexity of the training process in our proposed incremental learning is:

$$O(\|IS_{sv}\cup NS\|\times\|IS_{sv}\|^2)$$

Because our incremental learning method only retrains the classifier with the SV set and those important samples and $\|NS\|<<\|INS\|$, our proposed incremental learning is faster compared with both the repetitive learning method and the method proposed by Syed et al[6]. It is also faster than the method proposed by Xiao et al[7].

In the previous discussion we only focus on binary-classification problem. But real application problems are usually multi-classed. Over the past few years, a lot of progress has been made to construct multi-class classifiers based on the SVM theory. Besides the two classical structures, i.e., one-against-all and one-against-one, Platt et al[12] adopt the Decision Directed Acyclic Graph (DDAG) to combine many binary (two-class) classifiers into a multi-class classifier. Basically, the algorithm is based on binary classifiers and it is similar to the one-against-one structure. Dietterich et al[13] and Allwein et al[14] also construct multi-class classifier by combining the outputs of several binary ones. Typically, the combination is done via a simple nearest-neighbor rule, which finds the class that is the closest in some sense to the outputs of the binary classifiers[15]. Weston et al[11] have proposed an extension to solve multi-class classification problems in one step. But they cannot get a better result than the two classical structures in experiments. The one-against-all structure is the most commonly used multi-class classification utilities. However, the one-against-one structure shows better efficiency than the one-against-all structure in incremental learning, especially for a large set of samples with a small number of classes. In this paper, we will use two classical structures (one-against-all and one-against-one) to build multi-class classifiers for our application-sketch

recognition.

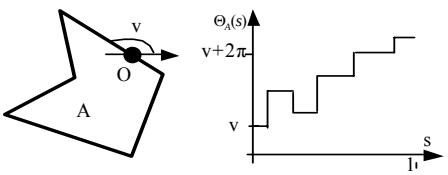
Define the training sets as consisting of m classes and n samples. For one-against-all structure, m sub-classifiers are needed, with n samples for each classifier. For one-against-one structure, $m(m\text{-}1)/2$ sub-classifiers are needed. Based on the max-win scheme[14], for the sample to be classified, each sub-classifier casts one vote for its preferred class. The final result is the class with the most votes. Compared with traditional one-against-all structure, one-against-one has the following advantages: firstly, although intuitively simple, one-against-one structure actually controls the Vapnik-Chervonenkis (VC)[5] dimensions on the whole training set, while the decision planes in one-against-all structure are always too complicated to avoid over-fitting. Secondly, in the one-against-all structure each classifier has to be trained using all the n samples. It is very time-consuming compared with the one-against-one structure. Finally, compared with one-against-all structure, one-against-one structure is more suitable for incremental learning. That is, if a new sample is added, for one-against-all structure we need to train each of the m classifier with n+1 samples; but for one-against-one structure we only need to train m-1 classifier each with $2(n+1)/m$ samples averagely. Hence we draw a conclusion that one-against-one structure is more suitable than one-against-all structure for incremental learning.

3. Experiments for Sketchy Shape Recognition

In order to validate our proposed approach, we applied all the algorithms in a real application[3] We construct classifiers based on repetitive learning and the three SVM-based incremental learning algorithms for classifying the sketchy shapes drawn by users and compare their performance.

We use turning function[10] to extract stroke features. Turning function is a function of arc s on a polygon A, as shown in Figure 1. It represents the cumulative turning angle from a start point on the boundary of A to the current point. In order to make sure the sketches are independent of size and location, we need to normalize the sketches before feature extraction. Meanwhile, we add some restrictions to turning function to simplify the problem and thus modify the turning function as

$$V_i = \Theta'_A(\frac{i \bmod d}{d}) - \Theta'_A(\frac{i-1}{d}), 0 \le i \le d$$



(a) Polygon (b) turning function

Figure 1 Turning Function

But it is mainly used to extract single-stroke features. For multi-stroke sketches, we introduce the virtual strokes to link the end of one stroke with the start of the next one orderly and continuously, as shown in Figure 2. In this way, we can transform a multi-stroke sketch to a single-stroke one. Then we can use turning function mentioned above to extract the stroke features.

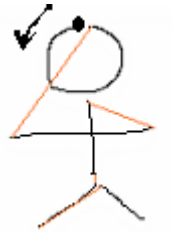


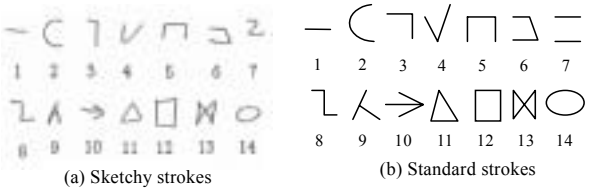
Figure 2 Turning Function

Four algorithms are compared in the experiment: the repetitive learning algorithm, Syed et al.'s and Xiao et al.'s incremental learning algorithm, and our proposed incremental learning algorithm. We collected two datasets, a 5-class dataset and a 14-class dataset[16][17]. The 5-class dataset is used to validate user adaptation of our proposed algorithm. The 14-class dataset together with the 5-class dataset are used to evaluate and compare the performance of the four algorithms. In addition, the 14-class dataset is used to compare the performance of the four algorithms with two different multi-class structures: one-against-one and one-against-all.

In order to collect the 5-class dataset, we asked a user to draw each shape (triangle, quadrangle, pentagon, hexagon, and ellipse) repeatedly with two different styles: one using the mouse and the other using the pen/tablet. In total, we have collected 1367 sample shapes drawn with pen/tablet and 325 samplers drawn with mouse. We use the turning function[8] as the feature vector of the polygon representation of the sketchy shape. In this paper, we use a 20-dimension feature vector and after transformation we can have 40 samples for each sample. Therefore we have 54680 samples for the pen/tablet style and 13000 samples for the mouse style in total. We randomly select 20210 samples of the pen/tablet to form a test set, TS_1, and use 12000 samples of the mouse style to form another test set, TS_2. Then, we randomly select samples from the remained samples of the pen/tablet style to form 39 incremental training sample sets (denoted as: IS_1, IS_2… IS_{39}). The first 6 incremental training sets have 100, 100, 120, 150, 300, 700 samples, respectively, and each of the rest 33 sets has 1000 samples. The remained samples of the mouse style form a training set with 1000 samples (denoted as IS_{40}) and we add this dataset into the incremental learning process as the last training set. Specially, for Xiao et al.'s algorithm, we use IS_1+IS_2+IS_3 as the first training set because it cannot work well (cannot terminate) if the training set is too small.

For the 14-class dataset, we collected 52802 samples of 14 classes of commonly used strokes. The sketchy shapes and their regular shapes for the 14 classes of strokes are shown in Figure 3. Each sample is

represented using a 20-dimentional features vector based on the turning function. We randomly select 21932 samples to constitute the test set (TS_3) and then randomly select samples from the remaining samples to form 24 incremental training sample sets (denoted as S_1, S_2… S_{24}). The first 5 incremental training sets have 300, 300, 370, 500, and 900 samples, separately, and each of the rest 19 sets has 1500 samples, respectively. Specially, for Xiao et al.'s algorithm, we use S_1+S_2+S_3 and S_4+S_5 as the first two training sets because it cannot work well for small training sets.



(a) Sketchy strokes (b) Standard strokes

Figure 3 Strokes of 14 classes frequently used in sketching

In all our experiments, a RBF (Radial Basic Function[5]) kernel is used and the implementation of the training process *Train*() is the same as in SVMTorch[18]. The parameter we used is $\delta=\sqrt{2}$, which can yield best performance after many tests. In all comparison, the discarding rate of Xiao et al.'s algorithm is 0.5. All experiments are done on an Intel Celeron PC (with a 2.4G Hz CPU and 256MB memory) running on Microsoft Windows XP Professional.

4. Performance Evaluation

We have done some experiments to evaluate and compare the performance of the four algorithms with both the 5-class dataset and the 14-class dataset. In the experiment on the 14-class dataset, we also compare the performance of one-against-one and one-against-all multi-class structures, with all four incremental learning algorithms.

Figure 4 shows the performance comparison among the four algorithms using the same one-against-one classifier structure on the 5-class dataset. For each algorithm, we use the same 40 incremental training sample sets to train them incrementally and record the incremental training time in Figure 4(c). After each incremental training step, we test it using the two test sets, TS_1 and TS_2, respectively, and record their classification precisions. Figure 4(a) and (b) show the curves of the precisions. Figure 5 shows the performance comparison among the four algorithms using two different classifier structures on the 14-class dataset. For each structure, we use the same 24 incremental training sample sets to train them incrementally and record the incremental training time in Figure 5(b) and (d). We also test their classification precisions on the test set (TS_3) and show them in Figure 5(a) and (c).

Theoretically, incremental learning is much faster than repetitive learning. Figure 4(c) and Figure 5(b)(d) have proved this point. Repetitive learning is much more time-consuming than the other three incremental learning algorithms, and its training time increases sharply as the scale of the training set increases. The training time of Xiao et al.'s algorithm is always fluctuant and Syed et al.'s algorithm shows a better effect in the training time. Compared with the other methods, our proposed incremental learning algorithm is the fastest, as shown in Figure 4(c) and Figure 5(b)(d). From these figures, we can see that the recognition precisions of all four algorithms are nearly close. The loss in precision of our incremental learning algorithm is very small. From Figure 4, we can see that the loss is only 0.75% (on TS_1) and 0.37% (on TS_2) compared with the repetitive learning algorithm, 0.4% (on TS_1) and 0.15% (on TS_2) compared with Syed et al.'s algorithm. Our algorithm outperforms Xiao et al.'s algorithm in most cases.

We also did some experiments to compare the two structures of multi-class classifiers using all the four SVM-based learning algorithms. From the experiments, as shown in Figure 6, we can see the one-against-one structure classifiers outperform the one-against-all structure classifiers in both training time and classification precision. This is why we finally adopt the one-against-one structure in our system.

In our on-line graphics recognition system, we have realized a multi-class SVM classifier based on the one-against-one structure. Each sub-classifier is based on our proposed incremental learning algorithm. Although the initial precision (95.53%) is very high in a general sense, it still may not be suitable for a specific user's drawing style. Users can correct these recognition errors through the UI of our system[3] immediately and the system will keep all these samples. After several samples are corrected, users can command the system to do incremental learning on these newly obtained samples. After training for less than 30 seconds, the new classifier is adaptive to the user's drawing style. Now, the system can correctly recognize them without making any previous mistakes.

5. Conclusion

Adaptation is a critical problem in user-specific freehand sketch recognition systems. SVM-based incremental learning is a good solution to this problem. Compared with repetitive learning, incremental learning can save much time in re-training with little or no loss of recognition precision. By utilizing the respective advantages of the two existing SVM-based incremental learning algorithms[6][7], we proposed a SVM-based incremental learning algorithm. Our proposed incremental learning algorithm can reduce both the training time and the required storage space for the training dataset to a large extent with very little loss of precision. Experiment results have shown that this algorithm is effective for user adaptation in on-line sketch recognition systems.

Nevertheless, the computational model of SVM

classifier must be parameterized. This means that it can only deal with single strokes. So, we made a supposition that all composite shapes are composed of some continuous strokes, and introduce the virtual strokes to link these continuous strokes orderly and translate the multi-strokes of composite shapes into a single stroke. The main problem of this method is that the strokes are not the basic constitutive geometric primitives of a shape for human cognition and the shape representation based on a single stroke composed of physical stokes and virtual strokes is not structural and unique for user. In fact, the underlying factors that determine the true identity of freehand sketches remain to be intractable and it is probably safe to say that no simple scheme is likely to achieve high recognition and reliability rates not to mention human performance. So, we must do online sketchy shape recognition based on a dynamic user modeling, which would model user's physical drawing contexts incrementally and user's subjective evaluations of the recognized objects accumulatively. The goal is to cope with the variations and ambiguities inherent in sketchy drawings so as to interpret the visual scene as the way the user intended. These are our ongoing research direction.

Acknowledgement

The work described in this paper was supported by a grant from National Natural Science Foundation of China (69903006, 60373065 and 60473113).

References

[1] Levent Burak Kara, Thomas F, Stahovich, Sim-U-Sketch: A Sketch-Based Interface for Simulink, Proceedings of AVI-2004: 354-357.

[2] Newman M W, James L, Hong J I, et al, DENIM: An informal web site design tool inspired by observations of practice, HCI, 2003(18): 259-324.

[3] Sun Z X, Xu X G, Sun J Y et al, Sketch-based Graphic Input Tool for Conceptual Design, J. of Computer-Aided Design & Computer Graphics (In Chinese), 2003,15(9):205~206.

[4] Xu X G, Sun Z X, Peng B B, et al, An online composite graphics recognition approach based on matching of spatial relation graphs, International Journal of Document Analysis and Recognition, 2004,7(1): 45-55.

[5] Vapnik V, The Nature of Statistical Learning Theory, Springer-Verlag, 1995.

[6] Syed N, Liu H and Sung K K, Incremental Learning with Support Vector Machines, Proc. of IJCAI-99, 1999.

[7] Xiao R, Wang J C, Sun Z X et al, An Incremental SVM Learning Algorithm α–ISVM, Journal of Software (In Chinese), 2001, 12(12): 1818-1824.

[8] Sun Z X, Peng B B, Cong L L, et al, Study on user adaptation for online sketchy graphic recognition, J. of Computer-Aided Design & Computer Graphics (In Chinese), 2004,16(9): 1207-1215.

[9] Sun Z X, Liu W Y, Peng B B, et al, User Adaptation for Online Sketchy Shape Recognition, in: J. Lladós, Y.B. Kwon (eds), Graphics Recognition: Recent Advances and Perspectives (Revised Papers from 5th International Workshop, GREC 2003, Barcelona, Catalonia, Spain 2003), Lecture Notes in Computer Science, 2004, Vol 3088: 303-314。

[10] Esther, M. A., An Efficient Computable Metric for Comparing Polygonal Shapes, IEEE Trans. on Pattern Analysis and Machine Intelligence, 1991, 13(3): 209~216。

[11] Weston J and Watkins C, Multi-class Support Vector Machines, Technical Report CSD-TR-98-04, Department of Computer Science, University of London, 1998.

[12] Platt J, Cristianini N and Taylor S, Large Margin DAGS for Multiclass Classification, Advances in Neural Information Processing Systems 12, eds. Solla S A, Leen T K, and Muller K R, MIT Press, 2000.

[13] Dietterich T and Bakiri G, Solving Multi-class Learning Problems via Error-Correcting Output Codes, Journal of Artificial Intelligence Research, 1995,2: 263-286.

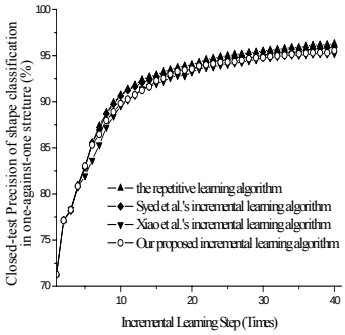
[14] Allwein E, Schapire R and Singer Y, Reducing Multiclass to Binary: A Unifying Approach for Margin Classifiers, Journal of Machine Learning Research, 2000(1, 2): 113-141.

[15] Klautau A, Jevtić N and Orilitsky A, On Nearest-Neighbor Error-Correcting Output Codes with Application to All-Pairs Multiclass Support Vector Machines, *Journal of Machine Learning Research*, 2003(4): 1-15.

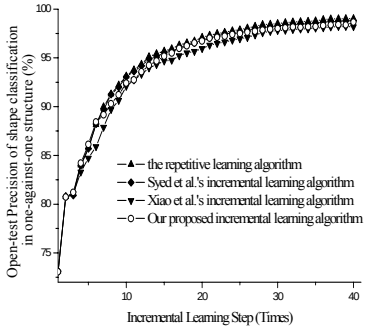
[16] Peng B B, User Adaptation for On-Line Sketchy Shape Recognition, Ms Degree Thesis ((In Chinese)), Nanjing University,2003.

[17] Sun J Y, User Adaptation for On-Line Sketchy Shape Recognition, Ms Degree Thesis ((In Chinese)), Nanjing University,2004.

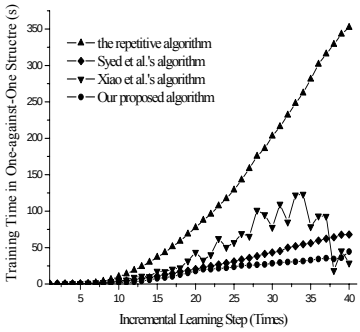
[18] Collobert R and Bengio S, SVMtorch: Support Vector Machines for Large-Scale Regression Problems, Journal of Machine Learning Research, 2001:143-160.



(a) Precision comparison (on TS_1)

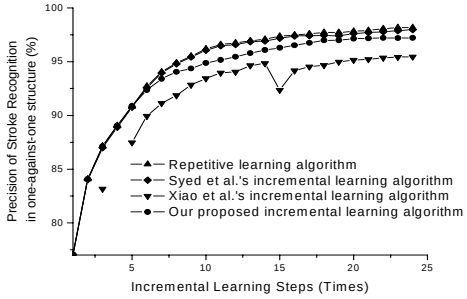


(b) Precision comparison (on TS_2)

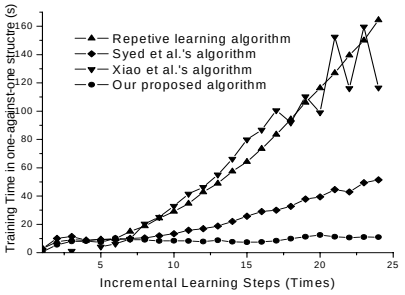


(c) Training time comparison

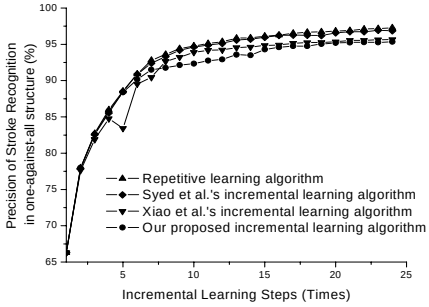
Figure 4. Performance evaluation among the four algorithms using one-against-one classifier structure on the 5-class dataset ($\delta = \sqrt{2}$ for the kernel functions of all four algorithms, and the discarding rate of Xiao et al.'s algorithm is 0.5).



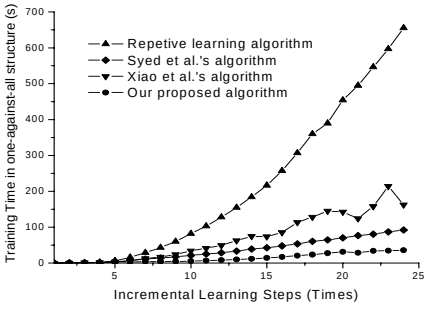
(a) Precision comparison using 1-1 structure



(b) Training time comparison using 1-1 structure

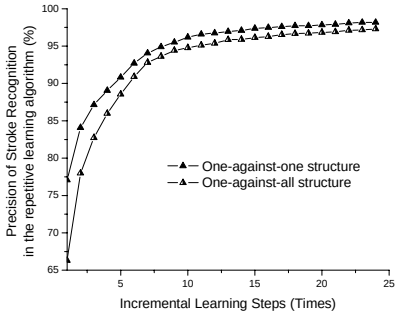


(c) Precision comparison using 1-m structure

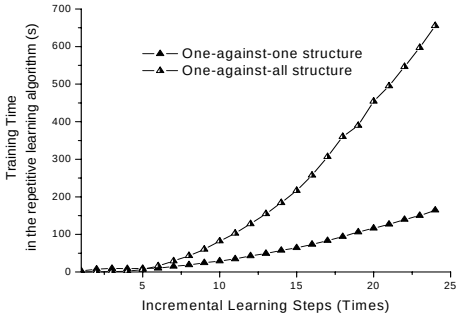


(d) Training time comparison using 1-m structure

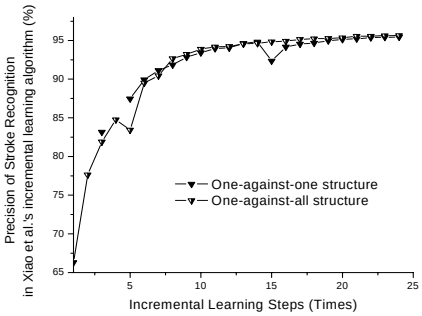
Figure 5. Performance evaluation among the four algorithms using different multi-class classifier structures on the 14-class dataset



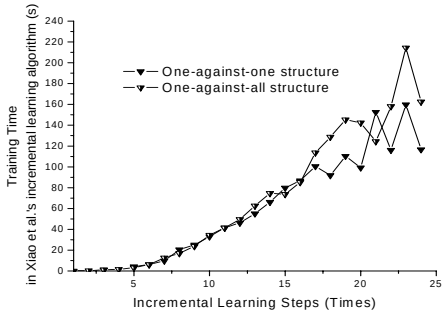
(a) Precision comparison using the repetitive learning algorithm



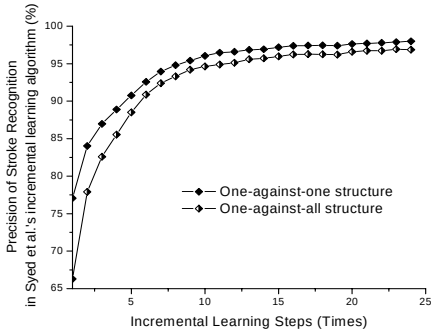
(b) Training time comparison using the repetitive learning algorithm



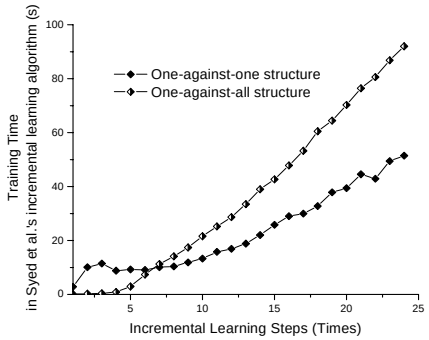
(c) Precision comparison using Xiao et al's algorithm



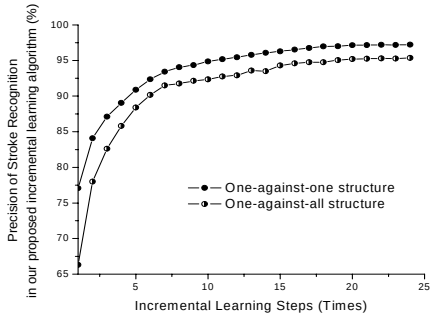
(d) Training time comparison using Xiao et al's algorithm



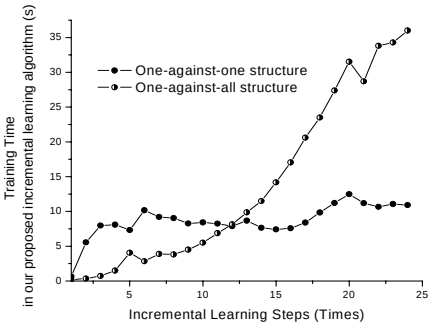
(e) Precision comparison using Syed et al's algorithm



(f) Training time comparison using Syed et al's algorithm



(g) Precision comparison using our proposed algorithm



(h) Training time comparison using our proposed algorithm

Figure 6. Precision and training time comparison among the two different structures (using all the four algorithms)