Fifth International Workshop on Computational Intelligence & Applications
IEEE SMC Hiroshima Chapter, Hiroshima University, Japan, November 10, 11 & 12, 2009

A Clustering-Based Algorithm for Data Reduction

Chi-Yuan Yeh, Jeng Ouyang, and Shie-Jue Lee
Department of Electrical Engineering
National Sun Yat-Sen University
Kaohsiung 80424, Taiwan
leesj@mail.ee.nsysu.edu.tw

***Abstract*—Finding an efficient data reduction method for large-scale problems is an imperative task. In this paper, we propose a similarity-based self-constructing fuzzy clustering algorithm to do the sampling of instances for the classification task. Instances that are similar to each other are grouped into the same cluster. When all the instances have been fed in, a number of clusters are formed automatically. Then the statistical mean for each cluster will be regarded as representing all the instances covered in the cluster. This approach has two advantages. One is that it can be faster and uses less storage memory. The other is that the number of new representative instances need not be specified in advance by the user. Experiments on real-world datasets show that our method can run faster and obtain better reduction rate than other methods.**

***Index Terms*—Large-scale dataset, fuzzy similarity, data reduction, prototype reduction, instance-filtering, instance-abstraction.**

I. Introduction

Due to rapid progress of information technologies, Internet technologies, and advanced database system technologies, the amount of data has been growing explosively, such as, text mining datasets, web mining datasets, image mining datasets, network auditing datasets, etc. A classifier generally suffers from very large memory requirements, and may result in very slow execution speed or even failure. Therefore, finding an efficient method to reduce the demand in time complexity and storage requirements, without degrading the generalization accuracy, is imperative. Data reduction techniques, also called prototype reduction techniques, which avoid using all the training data, can be applied for this purpose.

The data reduction techniques can be divided into two categories. One is instance-filtering which selects a part of original instances from the training data as representative instances. The other is instance-abstraction which generates new instances by summarizing the characteristics of similar instances as representative instances. Usually, the two approaches are used independently. Lam *et al.* [1] integrated the advantages of instance-filtering and instance-abstraction approaches to develop a framework, called prototype generation and filtering (PGF), for data reduction. They claimed that PGF works well and gains a significant improvement in data reduction compared with pure filtering-based and pure abstraction-based approaches. In the literature, most of the data reduction techniques were developed for instance-based classifiers (i.e., k-nearest neighbor [2]) to speed up searching the nearest neighbors [1, 3–13]. These approaches suffer from excessive computational burden for large-scale problems. To overcome this problem, Kim and Oommen [14] proposed an adaptive recursive partitioning algorithm, in which the training data are recursively subdivided into smaller subsets by the k-means algorithm [15] to filter out the less useful internal instances, and then apply conventional data reduction techniques to process the smaller subsets of the training data.

The simplest way of instance-filtering is random sampling which selects at random a small portion of the training data as new representative instances. Random sampling by SVM [16–18] classifiers usually has better relative performance at higher data reduction rates [19–22]. However, this approach may fail to work due to the highly unbalanced training data. Stratified sampling which selects a small portion of the training data per class uniformly at random can be used and slightly outperforms random sampling [23]. Lee and Huang [24] also claimed that random sampling can be further improved by stratified sampling. Besides, they implemented uniform random subset selection which is a space-filling design for SVM and claimed that uniform random subset selection is an optimal sampling strategy. Instance-abstraction approaches include the symbolic nearest mean classifier [6] in which k-means approach [15] is adopted to group similar instances of the same class and generates new representative instances using cluster means. Lozano *et al.* implemented learning vector quantization (LVQ) [25] and found that LVQ with dissimilarity-based classifier works well. Sanchez [10] proposed three abstraction-based approaches, called RSP-1, RSP-2, and RSP-3, based on space partitioning for data reduction. Lam *et al.* [1] adopted agglomerative clustering approach which merges two instances with the shortest distance to form a new instance at each iteration for instance abstraction.

We propose an instance-abstraction based approach called self-constructing fuzzy clustering (SCFC) algorithm to reduce the number of instances for the classification task. The proposed algorithm is an incremental-based approach in which one instance is considered at a time. Moreover, a similarity measure is developed to judge whether an instance and a cluster are similar. Each cluster is characterized by a membership function with statistical mean and standard deviation. Instances that are similar to each other are grouped into the same cluster. Once an instance is combined into an existing cluster, the membership function of that cluster should be updated. If an instance is not similar to any existing cluster, a new cluster is created for this instance. When all the instances have been fed in, a desired number of clusters are formed automatically; thus, the user need not specify the number of representative instances in advance, and trial-and-error for determining the appropriate number of representative instances can then be avoided. We then have one new representative instance for each cluster. Experiments on real-world datasets show that our method can run faster and obtain better reduction rate than other methods.

The rest of this paper is organized as follows. Section II presents the proposed similarity-based self-constructing fuzzy clustering algorithm. An example illustrating how the algorithm works is given in Section III. Experimental results are presented in Section IV. Finally, we conclude this work in Section V.

II. Proposed Method

Most of the existing data reduction techniques suffer from the problem of excessive computational burden by calculating the distance between any two instances on the training data. To overcome this problem, we propose a similarity-based self-constructing fuzzy clustering algorithm which is an incremental-based approach to reduce the number of instances for the classification task. The proposed approach has two advantages. One is that it can be faster and use less storage memory. The other is that the number of new representative instances need not be specified in advance by the user. Moreover, when calculating similarity, the variance of the underlying cluster is taken into consideration.

A. Self-Constructing Fuzzy Clustering

The basic concept of the self-constructing fuzzy clustering (SCFC) algorithm is that one training instance is considered at a time, and then the similarity measures between this instance and the existing fuzzy clusters are calculated to determine either to assign it to the most similar cluster or to create a new cluster for it. Once the training instance is combined into an existing cluster, the membership function of that cluster should be updated accordingly. On the contrary, when a new cluster is created, the corresponding membership function should be initialized. Our proposed SCFC is a five-layer network structure. When all the training instances have been fed in, a desired number of clusters are formed automatically.

Suppose we are given ℓ labeled training data $S_{tr} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), ..., (\mathbf{x}_\ell, y_\ell)\}$, where $\mathbf{x}_p = [x_{p,1}, x_{p,2}, ..., x_{p,n_1}, x_{p,n_1+1}, ..., x_{p,n}]$, $p = 1, 2, ..., \ell$, n_1 is the number of continuous features, n is the total number of features, ℓ is the number of training instances, $y_p \in \{1, 2, ..., k\}$, and k is the number of classes. Note that $[x_{p,1}, x_{p,2}, ..., x_{p,n_1}]$ is continuous part of $\mathbf{x}_p$; $[x_{p,n_1+1}, x_{p,n_1+2}, ..., x_{p,n}]$ is discrete part of $\mathbf{x}_p$. The clusters are $\mathbf{c}_1, \mathbf{c}_2, ..., \mathbf{c}_J$, respectively. Each cluster $\mathbf{c}_j$ has mean $\mathbf{m}_j = [m_{j,1}, m_{j,2}, ..., m_{j,n_1}, m_{j,n_1+1}, ..., m_n]$ and standard deviation $\boldsymbol{\sigma}_j = [\sigma_{j,1}, \sigma_{j,2}, ..., \sigma_{j,n_1}]$. Let s_j be the size of cluster $\mathbf{c}_j$. Initially, we have $J = 0$; namely, no clusters exist at the beginning. Then, the first cluster $\mathbf{c}_1$ is created with the first instance, i.e., $\mathbf{m}_1 = \mathbf{x}_1$, $y_{\mathbf{c}_1} = y_1$, $\boldsymbol{\sigma}_1 = \boldsymbol{\sigma}_0$, and $s_1 = 1$, where $\boldsymbol{\sigma}_0 = [\sigma_{0,1}, \sigma_{0,2}, ..., \sigma_{0,n_1}]$ is a user-defined constant vector. Now, we have $J = 1$.

Layer 1. This layer performs the fuzzification operation. Each node represents a cluster. Every node in this layer is an adaptive node. Note that if the training pattern S_{tr} contains discrete features and continuous feature, the membership function should be considered separately. Gaussian functions are adopted for the continuous part because of their superiority over other functions in performance [26]. Fuzzy singleton whose support is a single point with a membership function of one is adopted for the discrete part. The output of this layer for the continuous part is defined as follows:

$$O_{j,i}^{(1)} = \exp\left[-\left(\frac{x_{p,i} - m_{j,i}}{\sigma_{j,i}}\right)^2\right] \tag{1}$$

$i = 1, 2, ..., n_1$, $j = 1, 2, ..., J$, $x_{p,i}$ is the continuous part of instance $\mathbf{x}_p$, and $m_{j,i}$ and $\sigma_{j,i}$ denote the mean and standard deviation, respectively, of dimension i in cluster $\mathbf{c}_j$. The outputs of this layer for discrete part is defined as follows:

$$O_{j,i}^{(1)} = \begin{cases} 1 & \text{if } x_{p,i} = m_{j,i} \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

where $i = n_1 + 1, n_1 + 2, ..., n$, $j = 1, 2, ..., J$, $x_{p,i}$ is the discrete part of instance $\mathbf{x}_p$, and $m_{j,i}$ is the discrete part of cluster $\mathbf{c}_j$.

Layer 2. Every node in this layer is a fixed node labeled Π, whose output is the product of all the incoming signals:

$$O_j^{(2)} = \prod_{i=1}^{n} O_{j,i}^{(1)} \tag{3}$$

where $j = 1, 2, ..., J$. Each node output represents the input similarity of $\mathbf{x}_p$ to an existing cluster $\mathbf{c}_j$.

Layer 3. The single node in this layer is a fixed node labeled C, which performs competitive operation and the output is the largest input similarity, i.e.,

$$O^{(3)} = \max_{1 \le j \le J} O_j^{(2)}. \tag{4}$$

Let cluster $\mathbf{c}_a$ be the winner cluster, i.e.,

$$a = \arg\max_{1 \le j \le J} O_j^{(2)}. \tag{5}$$

Layer 4. The single node in this layer is a fixed node, whose output represents the input-output similarity defined as follows:

$$O^{(4)} = O^{(3)} \times \delta\left(y_p, y_{\mathbf{c}_a}\right) \tag{6}$$

where the Kronecker delta function $\delta(\cdot, \cdot)$ is defined as follows:

$$\delta\left(y_p, y_{\mathbf{c}_a}\right) = \begin{cases} 1 & \text{if } y_p = y_{\mathbf{c}_a} \\ 0 & \text{otherwise} \end{cases}. \tag{7}$$

Layer 5. The single node in this layer is a fixed node with a hard limit function

$$O^{(5)} = \text{hardlim}\left(O^{(4)}\right) = \begin{cases} 1 & \text{if } O^{(4)} \geq \rho \\ 0 & \text{otherwise} \end{cases} \quad (8)$$

where $0 \leq \rho \leq 1$ is a predefined threshold.
If $O^{(5)} = 1$, we say that $\mathbf{x}_p$ passes the input-output similarity test on winner cluster $\mathbf{c}_a$. In this case, we regard $\mathbf{x}_p$ to be most similar to cluster $\mathbf{c}_a$ and assign $\mathbf{x}_p$ to cluster $\mathbf{c}_a$. Besides, $\mathbf{m}_a$ and $\boldsymbol{\sigma}_a$ of cluster $\mathbf{c}_a$ should be modified to include $\mathbf{x}_p$ as its member. The modification to cluster $\mathbf{c}_a$ is defined as follows:

$$m_{a,i} = \frac{s_a \times m_{a,i} + x_{p,i}}{s_a + 1} \quad (9)$$

where $i = 1, 2, ..., n_1$,

$$\sigma_{a,i} = \sigma_{0,i} + \sqrt{\frac{((s_a)^2 - 1)(\sigma_{a,i} - \sigma_{0,i})^2 + s_a\left(m_{a,i} - x_{a,i}\right)^2}{s_a(s_a + 1)}}, \quad (10)$$

where $\sigma_{0,i}$ is a user-defined initial standard deviation, $i = 1, 2, ..., n_1$, and

$$s_a = s_a + 1. \quad (11)$$

Note that J and the discrete part of $\mathbf{m}_a$ is not changed in this case.
If $O^{(5)} = 0$, there are no existing fuzzy clusters on which $\mathbf{x}_p$ has passed the input-output similarity test. For this case, we assume that $\mathbf{x}_p$ is not similar enough to any existing cluster and a new fuzzy cluster $\mathbf{c}_{J+1}$ is created with

$$\mathbf{m}_{J+1} = \mathbf{x}_p, \quad \boldsymbol{\sigma}_{J+1} = \boldsymbol{\sigma}_0. \quad (12)$$

The instance $\mathbf{x}_p$ is assigned to cluster $\mathbf{c}_{J+1}$. Note that the size of cluster $\mathbf{c}_{J+1}$, s_{J+1}, should be initialized, i.e.,

$$s_{J+1} = 1 \quad (13)$$

and the number of clusters is increased by 1, i.e.,

$$J = J + 1. \quad (14)$$

The above process is iterated until all instances have been processed. Consequently, we have J clusters, i.e., new training dataset $S'_{tr} = \{(\mathbf{x}'_1, y_1), (\mathbf{x}'_2, y_2)..., (\mathbf{x}'_J, y_J)\}$ where $\mathbf{x}'_j$ is a centroid (mean) of cluster $\mathbf{c}_j$, $j = 1, 2, ..., J$. Note that the training instances in a cluster have a high degree of similarity to each other. Besides, when new training instances are considered, the existing clusters can be adjusted or new clusters can be created, without the necessity of generating the whole set of clusters from the scratch.

III. An Example

We give an example to illustrate how the proposed method works. Let S_{tr} be a simple training data, containing 10 instances $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2)..., (\mathbf{x}_{10}, y_{10})\}$ of two classes C_1 and C_2, with 3 discrete features and 5 continuous features, as follows:

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \mathbf{x}_3 \\ \mathbf{x}_4 \\ \mathbf{x}_5 \\ \mathbf{x}_6 \\ \mathbf{x}_7 \\ \mathbf{x}_8 \\ \mathbf{x}_9 \\ \mathbf{x}_{10} \end{bmatrix} = \begin{bmatrix} 0.82\ 0.69\ 0.73\ 0.92\ 0.58 & A\ D\ H \\ 0.78\ 0.82\ 0.75\ 0.88\ 0.63 & A\ D\ H \\ 0.83\ 0.70\ 0.71\ 0.93\ 0.65 & B\ E\ G \\ 0.48\ 0.39\ 0.31\ 0.51\ 0.25 & B\ E\ G \\ 0.81\ 0.68\ 0.76\ 0.87\ 0.62 & C\ E\ G \\ 0.25\ 0.31\ 0.41\ 0.19\ 0.28 & C\ E\ G \\ 0.50\ 0.45\ 0.22\ 0.35\ 0.62 & A\ F\ K \\ 0.52\ 0.47\ 0.25\ 0.33\ 0.65 & A\ F\ K \\ 0.53\ 0.49\ 0.18\ 0.37\ 0.63 & A\ F\ K \\ 0.27\ 0.30\ 0.40\ 0.20\ 0.30 & C\ E\ G \end{bmatrix},$$

$$\begin{aligned} \mathbf{y} &= \begin{bmatrix} y_1 & y_2 & y_3 & y_4 & y_5 & y_6 & y_7 & y_8 & y_9 & y_{10} \end{bmatrix}^T \\ &= \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 2 & 2 & 2 & 2 & 2 \end{bmatrix}^T. \end{aligned} \quad (15)$$

We run the proposed self-constructing fuzzy clustering algorithm, by setting $\rho = 0.8^5 = 0.328$ and $\boldsymbol{\sigma}_0 = [0.1, 0.1, 0.1, 0.1, 0.1]$. Firstly, the first instance $\mathbf{x}_1$ is fed in SCFC and obtain the first cluster $\mathbf{c}_1$; we have

$$\begin{aligned} \mathbf{m}_1 &= [\ 0.82\ 0.69\ 0.73\ 0.92\ 0.58\ \ A\ D\ H\ ], \\ y_{\mathbf{c}_1} &= 1, \\ \boldsymbol{\sigma}_1 &= [\ 0.1\ 0.1\ 0.1\ 0.1\ 0.1\ ]. \end{aligned}$$

Then, the second instance $\mathbf{x}_2$ is fed in SCFC.

Layer 1. The membership degree is

$$O^{(1)}_{1,i} = [\ 0.852\ 0.914\ 0.961\ 0.779\ 0.779\ \ 1\ 1\ 1\ ].$$

Layer 2. The input similarity is

$$\begin{aligned} O^{(2)}_1 &= 0.852 \times 0.914 \times 0.961 \times 0.779 \times 0.779 \\ &\quad \times 1 \times 1 \times 1 \\ &= 0.454. \end{aligned}$$

Layer 3. The winner cluster is $\mathbf{c}_1$ and

$$O^{(3)} = 0.454.$$

Layer 4. The input-output similarity is

$$O^{(4)} = 0.454 \times 1 = 0.454.$$

Layer 5. Because $O^{(4)} > \rho$, we have $O^{(5)} = 1$. Thus, $\mathbf{x}_2$ is assigned to cluster $\mathbf{c}_1$, and $\mathbf{m}_1$ and $\boldsymbol{\sigma}_1$ of cluster $\mathbf{c}_1$ should be modified as

$$\begin{aligned} \mathbf{m}_1 &= [\ 0.800\ 0.705\ 0.740\ 0.905\ 0.605\ \ A\ D\ H\ ], \\ \boldsymbol{\sigma}_1 &= [\ 0.128\ 0.121\ 0.114\ 0.135\ 0.135\ ]. \end{aligned}$$

When all the training instances in S_{tr} have been fed in, we obtain 6 clusters $\mathbf{c}_1$, $\mathbf{c}_2$, $\mathbf{c}_3$, $\mathbf{c}_4$, $\mathbf{c}_5$, and $\mathbf{c}_6$, which are shown in Table I. The transformed dataset S'_{tr} is shown as follows

$$\begin{aligned}
\mathbf{X} &= \left[\begin{array}{cccccc} \mathbf{x}'_1 & \mathbf{x}'_2 & \mathbf{x}'_3 & \mathbf{x}'_4 & \mathbf{x}'_5 & \mathbf{x}'_6 \end{array} \right]^T \\
&= \left[\begin{array}{l} 0.800\ 0.705\ 0.740\ 0.905\ 0.605\ \ A\ D\ H \\ 0.830\ 0.700\ 0.710\ 0.930\ 0.650\ \ B\ E\ G \\ 0.480\ 0.390\ 0.310\ 0.510\ 0.250\ \ B\ E\ G \\ 0.810\ 0.680\ 0.760\ 0.870\ 0.620\ \ C\ E\ G \\ 0.260\ 0.305\ 0.405\ 0.195\ 0.290\ \ C\ E\ G \\ 0.517\ 0.470\ 0.217\ 0.350\ 0.633\ \ A\ F\ K \end{array} \right], \\
\mathbf{y} &= \left[\begin{array}{cccccc} y_1 & y_2 & y_3 & y_4 & y_5 & y_6 \end{array} \right]^T \\
&= \left[\begin{array}{cccccc} 1 & 1 & 1 & 1 & 2 & 2 \end{array} \right]^T.
\end{aligned}$$

Based on S'_{tr}, a classifier can be built.

IV. Experimental Results

In this section, we present experimental results to show the effectiveness of the proposed self-constructing fuzzy clustering method. We compare the proposed method with two instance-filtering based data reduction methods: stratified sampling [23, 24] and DROP3 [7], and two instance-abstraction based data reduction methods: k-means [27–29] and RSP3 [10]. For convenience, the proposed method is abbreviated as SCFC and stratified sampling method is abbreviated as SS. We use support vectors machines (SVM) [16–18] as the classifier to test the effectiveness of the reduced datasets. Two well-known datasets, shuttle and 1999 KDD Cup, taken from the UCI Repository of Machine Learning Databases [30] and 1998 DARPA Intrusion Detection Evaluation Program administered by MIT Lincoln Labs [31], respectively, are used in the following experiments. The shuttle dataset contains 58,000 instances composed of 7 classes. Each instance consists of 9 numerical features which were then scaled to fit into a range from -1 through 1. The shuttle dataset has a skewed distribution and approximately 80% of the data belongs to class 1. We use 43,500 instances for training and the rest for testing. In the 1999 KDD dataset, 4898431 connection records are used as training data and 311029 connection records are used as testing data. Both training and testing data contain one normal network traffic (Normal) and four major attack categories: Denial-of-Service (DoS), Probing (Prob), User-toRoot (U2R), and Remote-to-Local (R2L). Each instance consists of 32 numerical features and 9 nominal features. The numerical features were then scaled to fit into a range from -1 through 1. This dataset has a skewed distribution, approximately 20% of the data belongs to class normal and approximately 80% of the data belongs to class DoS.

Stratified sampling (SS) and k-means need to know the number of representative instances in advance. The number of representative instances for each class is the same as that obtained by SCFC. Note that RSP3 and DROP3 do not need to know the number of representative instances in advance. The results are summarized in Table II and III in which reduction rates, execution time, and accuracy are listed. Because SS is sensitive to the randomly selected representative instances and k-means is sensitive to the initial randomly selected cluster centers, we conduct independently 10 times for each dataset. Each entry in the table represents the average result of the 10 runs except for "Original" column, "RSP3" column, and "DROP3" column. Apparently, SS method runs much faster than other methods because it requires no extra computation. However, it may select inappropriate instances and results in bad performance. For example, SS with 1-NN gets 96.86% in classification accuracy for the shuttle dataset. From Table II, we can see that SCFC runs much faster than k-means, RSP3, and DROP3 except for the SS method. For example, SCFC needs 9.25 seconds for about 280 representative instances, while k-means requires 58.45 seconds and RSP3 requires 2070.48 seconds. DROP3 requires 5389 seconds for 15614 representative instances. Although, RSP3 and DROP3 run significantly slow, the reduction rate of RSP3 (99.36%) is higher than that of DROP3 (41.12%). SCFC performs better than k-means, SS, and RSP3 in classification accuracy. DROP3 performs slightly better than SCFC, but the reduction rate of DROP3 (41.12%) is lower than that of SCFC (99.35%).

From Table III, we can see that SCFC runs much faster than k-means, RSP3, and DROP3 except for the SS the method. For example, SCFC needs 473.92 seconds for about 1470 representative instances, while k-means requires 12618.35 seconds and RSP3 requires 218208.58 seconds. DROP3 requires 157795.70 seconds for 49403 representative instances. Although, RSP3 and DROP3 run significantly slow, the reduction rate of RSP3 (99.70%) is higher than that of DROP3 (90.00%). SCFC performs better than k-means, SS, and RSP3 in classification accuracy. DROP3 (92.41%, 92.51%, 92.50%) performs slightly better than that of SCFC (92.51%, 92.83%, 92.11%), but the reduction rate of DROP3 (90.00%) is lower than SCFC (99.70%). SS can not perform well, for example, SS with SVM gets 76.05% in classification accuracy for 1999 KDD Cup dataset. Note that the proposed SCFC can handle large-scale problem. We try to reduce the original 1999 KDD Cup dataset which contains 4,898,431 instances. SCFC spends 2048.66 seconds to abstract 3152 representative instances as shown in Table IV. The reduced instances get 92.38% in accuracy for SVM.

From Table IV, we can see that SCFC can handle the problem when pattern distributions are significantly different among different classes. Suppose one class contains patterns with a dense distribution, SCFC abstracts a small number of representative instances. On the contrary, if another class contains sparsely distributed patterns, SCFC abstracts a large number of representative instances. For example, class 1 (normal network traffic) in 1999 KDD Cup dataset is a sparse distribution, the reduction rate is 99.17%, while class 2 (DoS attack) is a dense distribution, the reduction rate is 99.91%. An interesting phenomenon in 1999 KDD Cup dataset is that the original size of class 2 is larger than that of class 1, but the reduced size of class 2 is smaller than that of class 1. However, this situation won't affect the classification accuracy.

In summary, SS method runs much faster than other methods because it requires no extra computation, but the selected instances are not good for classification. SCFC can not only run much faster than k-means, RSP3, and DROP3 in data reduction, but also provide comparably good or better abstracted new instances for classification. RSP3 has the same reduction rate as SCFC, but spends more time for calculating distances between instances. DROP3 works well in classification, but spends more time in data reduction. Besides, k-means is sensitive to the initial randomly selected cluster centers.

V. Conclusion

We have presented a self-constructing fuzzy cluster (SCFC) algorithm to reduce the number of instances for classification. We have compared SCFC with two instance-filtering based data reduction methods, stratified sampling and DROP3, and two instance-abstraction based data reduction methods, k-means and RSP3. Experimental results from two well-known datasets have shown that SCFC can run faster and obtain better reduction rate than other methods.

Acknowledgment

This work was supported by the Ministry of Economic Affairs under the grant 98-EC-17-A-02-S2-0114.

References

[1] W. Lam, C.-K. Keung, and C. X. Ling, "Learning good-prototypes for classification using filtering and abstraction of instances," *Pattern Recognition*, vol. 35, no. 7, pp. 1491–1506, July 2002.

[2] T. M. Cover and P. E. Hart, "Nearest neighbor pattern classification," *IEEE Transactions on Information Theory*, vol. 13, no. 1, pp. 21–27, January 1967.

[3] D. L. Wilson, "Asymptotic properties of nearest neighbor rules using edited data," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 2, no. 3, pp. 408–421, July 1972.

[4] G. W. Gates, "The reduced nearest neighbor rule," *IEEE Transactions on Information Theory*, vol. 18, no. 3, pp. 431–433, May 1972.

[5] G. L. Ritter, H. B. Woodruff, S. R. Lowry, and T. L. Isenhour, "An algorithm for a selective nearest neighbor decision rule," *IEEE Transactions on Information Theory*, vol. 21, no. 6, pp. 665–669, November 1975.

[6] P. Datta and D. Kibler, "Symbolic nearest mean classifiers," in *Proceedings of the 14th International Conference on Machine Learning*, July 1997, pp. 82–87.

[7] D. R. Wilson and T. R. Martinez, "Reduction techniques for instance-based learning algorithms," *Machine Learning*, vol. 38, no. 3, pp. 257–286, March 2000.

[8] H. Brighton and C. Mellish, "Reduction techniques for instance-based learning algorithms," *Data Mining and Knowledge Discovery*, vol. 6, no. 2, pp. 153–172, April 2002.

[9] S.-W. Kim and B. J. Oommen, "Enhancing prototype reduction schemes with LVQ3-type algorithms," *Pattern Recognition*, vol. 36, no. 5, pp. 1083–1093, May 2003.

[10] J. Sánchez, "High training set size reduction by space partitioning and prototype abstraction," *Pattern Recognition*, vol. 37, no. 7, pp. 1561–1564, July 2004.

[11] M. Lozano, J. M. Sotoca, J. S. Sánchez, F. Pla, E. Pękalska, and R. P. W. Duin, "Experimental study on prototype optimisation algorithms for prototype-based classification in vector spaces," *Pattern Recognition*, vol. 39, no. 10, pp. 1827–1838, October 2006.

[12] S.-H. Son and J.-Y. Kim, "Data reduction for instance-based learning using entropy-based partitioning," in *Proceedings of the International Conference on Computational Science and its Applications*, May 2006, pp. 590–599.

[13] E. Marchiori, "Hit miss networks with applications to instance selection," *Journal of Machine Learning Research*, vol. 9, pp. 997–1017, June 2008.

[14] S.-W. Kim and B. J. Oommen, "Enhancing prototype reduction schemes with recursion: A method applicable for "large" data sets," *IEEE Transactions on Systems, Man, and Cybernetics, part B: Cybernetics*, vol. 34, no. 3, pp. 1384–1397, June 2004.

[15] J. Macqueen, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability*, 1967, pp. 281–297.

[16] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, September 1995.

[17] V. Vapnik, *The nature of statistical learning theory*, 2nd ed. New York, NY, USA: Springer, November 1999.

[18] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines and other kernel-based learning methods*. New York, NY, USA: Cambridge University Press, March 2000.

[19] N. A. Syed, H. Liu, and K. K. Sung, "A study of support vectors on model independent example selection," in *Proceedings of the 5th ACM SIGKDD international conference on Knowledge discovery and data mining*, August 1999, pp. 272–276.

[20] Y.-J. Lee and O. L. Mangasarian, "RSVM: Reduced support vector machines," in *Proceedings of the First SIAM International Conference on Data Mining*, April 2001, pp. 350–366.

[21] K.-M. Lin and C.-J. Lin, "A study on reduced support vector machines," *IEEE Transactions on Neural Networks*, vol. 14, no. 6, pp. 1449–1559, November 2003.

[22] J. G. Wang, P. Neskovic, and L. N. Cooper, "Training data selection for support vector machines," in *Proceedings of the 1st International Conference on Advances in Natural Computation*, August 2005, pp. 554–564.

[23] E. Pękalska, R. P. W. Duin, and P. Paclík, "Prototype selection for dissimilarity-based classifiers," *Pattern Recognition*, vol. 39, no. 2, pp. 189–208, February 2006.

[24] Y.-J. Lee and S.-Y. Huang, "Reduced support vector machines: A statistical theory," *IEEE Transactions on Neural Networks*, vol. 18, no. 1, pp. 1–13, January 2007.

[25] Y. Linde, A. Buzo, and R. Gray, "An algorithm for vector quantizer design," *IEEE Transaction on Communications*, vol. 28, no. 1, pp. 84–95, January 1980.

[26] G. J. Klir and B. Yuan, *Fuzzy sets and fuzzy logic: theory and applications*, 1st ed. Upper Saddle River, NJ, USA: Prentice Hall PTR, May 1995.

[27] M. B. de Almeida, A. de Padua Braga, and J. P. Braga, "SVM-KM: speeding SVMs learning with a priori cluster selection and k-means," in *Proceedings of the 6th Brazilian Symposium on Neural Networks*, November 2000, pp. 162–167.

[28] S. Zheng, X. Lu, N. Zheng, and W. Xu, "Unsupervised clustering based reduced support vector machines," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, April 2003, pp. 821–824.

[29] R. Koggalage and S. K. Halgamuge, "Reducing the number of training samples for fast support vector machine classification," *Neural Information Processing - Letters and Reviews*, vol. 2, no. 3, pp. 57–65, March 2004.

[30] A. Asuncion and D. Newman, "UCI machine learning repository," 2007. [Online]. Available: http://www.ics.uci.edu/∼mlearn/MLRepository.html

[31] S. J. Stolfo, W. Fan, W. Lee, A. Prodromidis, and P. K. Chan, "Cost-based modeling for fraud and intrusion detection: Results from the jam project," in *In Proceedings of the 2000 DARPA Information Survivability Conference and Exposition*, January 2000, pp. 130–144.

Table I
FOUR CLUSTERS OBTAINED FOR S_{tr}

	size s	mean	standard deviation
$\mathbf{c}_1$	2	$[$ 0.800 0.705 0.740 0.905 0.605 $A\ D\ H$ $]$	$[$ 0.128 0.121 0.114 0.135 0.135 $]$
$\mathbf{c}_2$	1	$[$ 0.830 0.700 0.710 0.930 0.650 $B\ E\ G$ $]$	$[$ 0.100 0.100 0.100 0.100 0.100 $]$
$\mathbf{c}_3$	1	$[$ 0.480 0.390 0.310 0.510 0.250 $B\ E\ G$ $]$	$[$ 0.100 0.100 0.100 0.100 0.100 $]$
$\mathbf{c}_4$	1	$[$ 0.810 0.680 0.760 0.870 0.620 $C\ E\ G$ $]$	$[$ 0.100 0.100 0.100 0.100 0.100 $]$
$\mathbf{c}_5$	2	$[$ 0.260 0.305 0.405 0.195 0.290 $C\ E\ G$ $]$	$[$ 0.114 0.107 0.107 0.107 0.114 $]$
$\mathbf{c}_6$	3	$[$ 0.517 0.470 0.217 0.350 0.633 $A\ F\ K$ $]$	$[$ 0.115 0.120 0.135 0.120 0.115 $]$

Table II
A COMPARISON OF DATA REDUCTION METHODS FOR SHUTTLE DATASET.

		Abstraction-based			Filtering-based	
	Original	SCFC	k-means	RSP3	SS	DROP3
Number of data	43500	284.4	280	279	280	15614
Reduction rate (%)	0.00	99.35	99.36	99.36	99.36	41.12
Time (sec)	0.00	9.25	58.45	2070.48	0.13	5389.00
SVM (%)	99.79	99.20	97.90	98.54	97.66	99.74

Table III
A COMPARISON OF DATA REDUCTION METHODS FOR 1999 KDD CUP DATASET.

		Abstraction-based			Filtering-based	
	Original	SCFC	k-means	RSP3	SS	DROP3
Number of data	494021	1470.8	1465	1555	1465	49403
Reduction rate (%)	0.00	99.70	99.70	99.69	99.70	90.00
Time (sec)	0.00	473.92	12618.35	218208.58	1.22	157795.70
SVM (%)	92.51	91.97	87.56	92.20	76.05	92.41

Table IV
DETAILED REDUCTION RESULTS FOR EACH DATASET.

	Shuttle		1999 KDD Cup		1999 KDD Cup (10%)	
	Original	Reduced	Original	Reduced	Original	Reduced
Class 1	34108	134	972781	1729	97278	805
Class 2	37	9	3883370	451	391458	351
Class 3	132	9	41102	879	4107	213
Class 4	6748	76	52	33	52	34
Class 5	2458	36	1126	60	1126	62
Class 6	6	58	-	-	-	-
Class 7	11	11	-	-	-	-
Total	43500	280	4898431	3152	494021	1465