International Journal of Innovative
Computing, Information and Control
Volume **8**, Number **6**, June **2012**

ICIC International ©2012 ISSN 1349-4198
pp. **3963–3980**

A NOVEL PROTOTYPE REDUCTION APPROACH FOR SUPERVISED LEARNING

Chih-Feng Liu, Chi-Yuan Yeh and Shie-Jue Lee*

Department of Electrical Engineering
National Sun Yat-sen University
No. 70, Lienhai Rd., Kaohsiung 80424, Taiwan
*Corresponding author: leesj@mail.ee.nsysu.edu.tw

Received February 2011; revised June 2011

Abstract. *We propose a similarity-based prototype reduction algorithm to reduce the training set size for supervised learning. Training patterns are input to the algorithm one by one and grouped into blobs through similarity tests. The statistical mean of each blob is regarded as a prototype representing all the patterns included in the blob. The collection of such means can then be used to substitute the original training set, and, consequently, the training set for later supervised learning is reduced. This approach has several advantages. The distribution of the data contained in each blob is statistically well described. Each obtained prototype is a good representative of the patterns included in the corresponding blob. Different numbers of representatives are extracted automatically according to the similarity relationship among and the distribution of the original training patterns. Furthermore, our method can be applied efficiently to both regression and classification problems. Experimental results show that the proposed method performs more effectively than other prototype reduction methods.*

Keywords: Large-scale dataset, Similarity measure, Reduction rate, Regression, Classification, Generalization accuracy

1. **Introduction.** Due to the rapid development of Internet and advanced database technologies, the amount of training data collected through text mining, web mining, image mining, or network auditing has been growing enormously. An automatic learning system, e.g., support vector machines and artificial neural networks, may require an unacceptably large amount of memory and execution time for a huge set of training data [1-4]. Therefore, finding an efficient method to reduce the training set size and thus reduce the complexity for learning, without degrading the generalization accuracy, is an important job. Prototype reduction techniques, also called data reduction techniques, which aim to avoid using all the patterns in the training dataset, are therefore very valuable [5] in machine learning.

Prototype reduction techniques can be divided into two categories. One is instance-filtering based which selects a subset of the original training dataset as representative prototypes. The other is instance-abstraction based which, by summarizing the characteristics of similar patterns, generates new prototypes to represent the original patterns of the training dataset. Most prototype reduction techniques have been developed for instance-based classifiers [5-13]. Kim and Oommen [11] proposed an adaptive recursive partitioning algorithm, in which the training dataset is recursively subdivided into smaller subsets to filter out less useful internal patterns, and then conventional prototype reduction techniques are applied to the resulting smaller subsets of the training data.

Random sampling is a simple way of instance-filtering, in which a small portion of the training data is chosen as representative patterns [14-17]. However, this approach may

fail to work well due to an uneven selection from the training data. Stratified sampling selects at random a small portion of the training data per class uniformly [18]. For support vector machines, Lee and Huang [19] implemented a uniform random subset selection with a space-filling design. Wilson and Martinez [9] proposed a decremental reduction optimization procedure for instance-based learning. Brighton and Mellish [20] proposed a two-stage method, called the iterative case filtering, for prototype reduction. Marchiori [13] proposed a graph-based method for prototype reduction. Three versions of this method were developed. Several instance-abstraction based prototype reduction approaches have also been proposed. Datta and Kibler [21] adopted the k-means algorithm [22] to group similar patterns into clusters and used the cluster means as representative patterns. Lozano et al. found that the learning vector quantization [23] worked well with dissimilarity-based classifiers. Sánchez [10] proposed three methods based on space partitioning for prototype reduction. Lam et al. [5] adopted an agglomerative clustering approach. Two nearest patterns are merged to form a new pattern at each iteration. Tabata et al. [24] proposed volume prototypes for streaming data. A volume prototype has a specific region which enables data compression based on local data distributions.

We propose a similarity-based prototype reduction method to reduce the number of training patterns for supervised learning. Training patterns are considered one by one and grouped into blobs. Each blob is characterized by a membership function having a statistical mean and standard deviation with it. A similarity measure is developed to judge whether an incoming pattern is similar to the members in an existing blob. If positive, the pattern is added into the existing blob and the membership function of the blob is updated. Otherwise, a new blob is created. When all the patterns have been considered, a desired number of blobs are formed automatically. The statistical mean of each blob is regarded as a prototype representing all the patterns included in the blob. The collection of such means is then used to substitute the original training dataset for later supervised learning. This approach has several advantages. The distribution of the data contained in each blob is statistically well described. Each obtained prototype is a good representative of the patterns included in the corresponding blob. Different numbers of representatives are extracted automatically according to the similarity relationship among and the distribution of the original training patterns. Furthermore, our method can be applied efficiently to both regression and classification problems. Experimental results show that the proposed method performs more effectively than other prototype reduction methods.

The rest of this paper is organized as follows. Section 2 gives a definition for the prototype reduction task. Section 3 gives a brief background of several other prototype reduction methods. Section 4 describes our proposed similarity-based prototype reduction algorithm, together with a brief analysis on complexity. Two examples illustrating how the algorithm works are given in Section 5. Some experimental results are presented in Section 6. Finally, we conclude this work in Section 7.

2. **Problem Definition.**

In a supervised learning problem, we are given a training set, S_{tr}, of ℓ labeled training patterns, i.e.,

$$S_{tr} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_\ell, y_\ell)\}, \tag{1}$$

where $\mathbf{x}_i = [x_{i,1}, x_{i,2}, \ldots, x_{i,n}]$ and y_i denote the n-dimensional input and the output, respectively, for pattern i, $i = 1, 2, \ldots, \ell$, and y_i and each element in $\mathbf{x}_i$ can be discrete or continuous values. If $y_i \in \{1, 2, \ldots, K\}$, the problem is a classification problem. In this case, we have K classes. We say $\mathbf{x}_i$ belongs to class y_i. On the other hand, if y_i is real, the problem is a regression problem. The patterns in S_{tr} can be divided into bins. For

classification, each class corresponds to one bin. For regression, the range of y_i values can be divided into intervals with each interval containing an equal number of patterns. Then each interval corresponds to one bin. Of course, other forms of bins are possible. For a training set S_{tr} with K bins, we use S_1, S_2, ..., and S_K to denote the subsets containing the training patterns included in bin 1, bin 2, ..., and bin K, respectively.

The prototype reduction task for supervised learning is to find a new training dataset S'_{tr}:

$$S'_{tr} = \{(\mathbf{x}'_1, y'_1), (\mathbf{x}'_2, y'_2), \ldots, (\mathbf{x}'_J, y'_J)\} \tag{2}$$

such that $J < \ell$ and S'_{tr} is sufficient for generalization. To be specific, the model characterizing the input-output relationship derived from S'_{tr} should be commensurable with that derived from S_{tr}. If J is much smaller than ℓ, the computation cost of finding the model from S'_{tr} can be drastically smaller than the computation cost of finding the model directly from S_{tr}. Some criteria can be used to measure the performance of a prototype reduction technique. Among them, reduction rate and precision are most obvious. The reduction rate is the ratio $(\ell - J)/\ell$. The measure of precision describes how well the reduced set S'_{tr} works for generalization. It is desirable to require both the reduction rate and precision be as high as possible. Another measure is the CPU time spent in deriving S'_{tr} from S_{tr}. Obviously, we'd like it to be as low as possible.

3. Related Work.

As mentioned earlier, prototype reduction techniques can be either instance-filtering based or instance-abstraction based. Some of them are briefly described below.

3.1. Instance-filtering based.

In case of instance-filtering, a subset of the original training dataset S_{tr} is selected to be the reduced training dataset S'_{tr}. One well known instance-filtering based method is stratified sampling (SS) [18,19] which requires the value J be specified by the user in advance. To get J patterns from S_{tr}, SS selects q_j training patterns uniformly at random from S_j such that

$$q_j = \frac{|S_j|}{|S_{tr}|} J \tag{3}$$

where $|\ |$ denotes cardinality and $J = q_1 + q_2 + \ldots + q_K$. This ensures that patterns from different bins are selected with equal probability.

DROP3 [9] is another instance-filtering based prototype reduction method which can only be used for classification. First, DROP3 removes the patterns which are misclassified by their k nearest neighbors. Next, the remaining patterns are sorted according to the distance to their nearest enemy. An enemy is a pattern of other classes. Then DROP3 removes one pattern at a time, in the order of decreasing distance, from the training set, if the removal does not deteriorate the classification accuracy induced. Another instance-filtering based prototype reduction method for classification only is the hit miss network (HMN) proposed by Marchiori [13]. Let x be the nearest neighbor of y. If x and y have the same class label, x has a hit. Otherwise, x has a miss. The hit and miss degrees of x represent the number of hits and misses, respectively, x has. There are three versions of this method, called HMN-C, HMN-E and HMN-EI, respectively. HMN-EI applies HMN-E iteratively to removing more redundant patterns, and was claimed to perform better than HMN-C and HMN-E.

3.2. Instance-abstraction based.

Using instance-abstraction, a new representative pattern is obtained by summarizing the characteristics of some of the original patterns. Therefore, in general, S'_{tr} is not a subset of S_{tr}. The k-means clustering algorithm [22] can be used for this purpose. For each S_j, $1 \leq j \leq K$, the user specifies the number of

clusters q_j to be obtained and an initial centroid for each cluster. Then all the patterns in S_j are partitioned into q_j clusters based on a distance measure. The centroids of the resulting q_j clusters are taken to be the representatives of all the patterns in S_j. Note that $q_1 + q_2 + \ldots + q_K = J$ and each q_j can be set by Equation (3).

Another instance-abstraction based prototype reduction method is RSP3 [10]. RSP3 performs partitioning until all the subsets are class homogeneous, i.e., the patterns in each subset belong to the same class [10]. Then one representative pattern is obtained for each subset. Initially, S_{tr} itself constitutes a subset. For any subset S that is not class homogeneous, RSP3 finds two patterns that are farthest apart in S. Let these two patterns be $\mathbf{x}_1$ and $\mathbf{x}_2$. Then S is separated into two subsets, one containing those patterns nearer to $\mathbf{x}_1$ and the other containing those patterns nearer to $\mathbf{x}_2$. This process continues until all the subsets are class homogeneous. Obviously, RSP3 can only be used for classification.

4. **Proposed Method.** The prototype reduction methods described in the previous section are problematic. Some, e.g., SS, perform reduction in a way without taking into account the characteristics inherently associated with the original training dataset. Some, e.g., k-means and DROP3, can only apply to classification problems. Some, e.g., DROP3 and RSP3, are very computation-demanding. We propose a similarity-based prototype reduction algorithm to deal with these problems. Training patterns are considered one by one and those similar to each other are grouped into the same blob through similarity tests. Blobs are identified based on the similarity relationship among the original training patterns. The statistical mean of each blob is regarded as a prototype representing all the patterns included in the blob. The collection of such means can then be used to substitute the original training set for later supervised learning.

4.1. **Similarity-based prototype reduction.** For convenience, we use $\mathbf{p}_i$ to denote $(\mathbf{x}_i, y_i)$, i.e.,

$$\begin{aligned}\mathbf{p}_i &= [p_{i,1}, p_{i,2}, \ldots, p_{i,m-1}, p_{i,m}] \\ &= (\mathbf{x}_i, y_i) = [x_{i,1}, x_{i,2}, \ldots, x_{i,n}, y_i]\end{aligned}$$

for $1 \leq i \leq \ell$, where $m = n + 1$. The similarity-based prototype reduction algorithm partitions the members of S_{tr} into blobs. Each blob is described by its mean $\boldsymbol{\mu}$, deviation $\boldsymbol{\sigma}$ and size s. An existing blob may be expanded to include new members and new blobs may be created during the reduction process. Let J be the number of existing blobs. Initially, no blobs exist, so J is zero. For a training pattern $\mathbf{p}_i = [p_{i,1}, p_{i,2}, \ldots, p_{i,m}]$ being considered, we compute the similarity of this pattern to each of the existing blobs. If $\mathbf{p}_i$ is not close enough to any existing blob, a new blob is created and J is increased by 1. Otherwise, $\mathbf{p}_i$ is added to the blob to which it is closest. In principle, our algorithm works like a five-layer network as shown in Figure 1. The five layers are referred to as the input layer, layer 1, layer 2, layer 3 and the output layer, respectively. The operation of each layer is described in detail below.

- Input layer. This layer transfers the m components of $\mathbf{p}_i$ forward to the nodes in layer 1. Node 1 of this layer passes $p_{i,1}$ to the first node of all the groups in layer 1, node 2 of this layer passes $p_{i,2}$ to the second node of all the groups in layer 1, etc.
- Layer 1. This layer contains J groups of nodes. J is the number of existing blobs. Each group has m nodes and corresponds to one existing blob. For example, the first group of m nodes are for blob 1, the second group of m nodes are for blob 2, etc. Each node computes as output the component membership degree of $\mathbf{p}_i$ in one dimension to the underlying blob. Therefore, $O_{j,1}^{(1)}$, $O_{j,2}^{(1)}$, $\ldots$, $O_{j,m}^{(1)}$ denote the

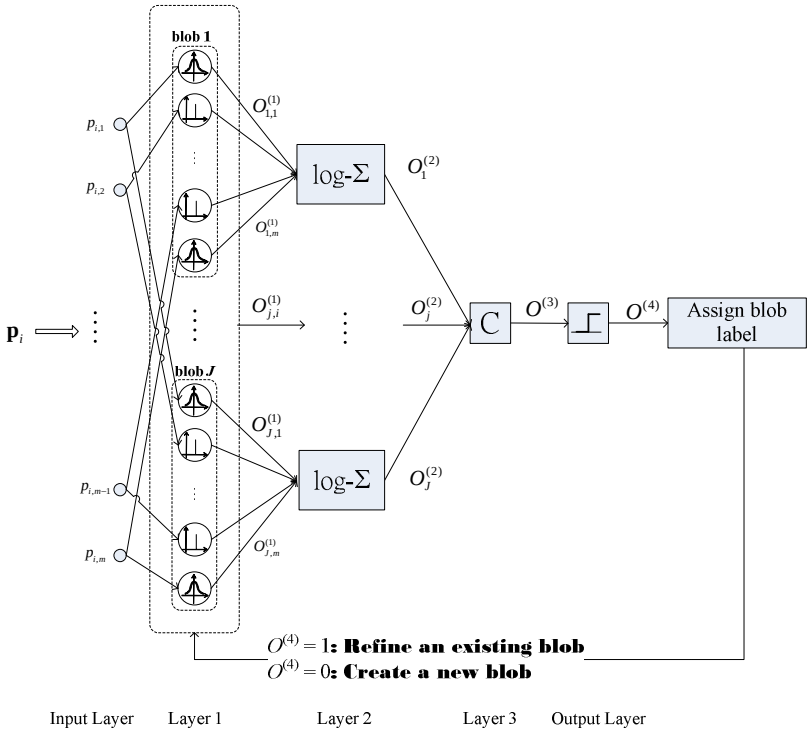


FIGURE 1. A five-layer network for generating blobs

component membership degrees of $\mathbf{p}_i$ to blob j for the first dimension, the second dimension, ..., and the mth dimension, respectively, and are defined by

$$O_{j,k}^{(1)} = \begin{cases} \exp\left[-\left(\dfrac{p_{i,k} - \mu_{j,k}}{\sigma_{j,k}}\right)^2\right], & \text{if component } k \text{ is continuous} \\ \delta\left(p_{i,k}, \mu_{j,k}\right), & \text{if component } k \text{ is discrete} \end{cases} \tag{4}$$

for $1 \leq k \leq m$ and $1 \leq j \leq J$, where $\boldsymbol{\mu}_j = [\mu_{j,1}, \ldots, \mu_{j,m}]$ and $\boldsymbol{\sigma}_j = [\sigma_{j,1}, \ldots, \sigma_{j,m}]$ are the mean and deviation, respectively, of blob j, and $\delta(x, y)$ is the Kronecker delta function defined as

$$\delta(x, y) = \begin{cases} 1, & \text{if } x = y \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

The Gaussian function is adopted for each continuous component, as shown in Equation (4), due to its superiority over other functions [25]. As usual, the power in Equation (4) is two. Its value has an effect on the number of blobs obtained. A larger value will make the boundary of the Gaussian function sharper and more blobs will be obtained in the end. Each component membership degree is a real number between 0 and 1. One condition imposed on the use of Equation (4) is that the blobs concerned cannot be slanted ones. A slanted blob may sometimes fit a set of patterns better than a non-slanted one. We'll investigate one improvement in Section 7. For discrete components, the component membership degree is 1 if the two involved values match and is 0 otherwise.

- Layer 2. This layer contains J nodes, each node corresponding to one existing blob. The output of each node in this layer denotes the similarity degree of $\mathbf{p}_i$ to the underlying blob, defined as the logarithm of the product of all the component

membership degrees, i.e.,

$$O_j^{(2)} = \log \prod_{k=1}^{m} O_{j,k}^{(1)} = \log O_{j,1}^{(1)} + \log O_{j,2}^{(1)} + \ldots + \log O_{j,m}^{(1)} \tag{6}$$

for $1 \leq j \leq J$. Taking the logarithm avoids the presence of underflow when m is large. We define $\log 0$ to be $-\infty$. A larger similarity degree indicates that $\mathbf{p}_i$ is more similar to the underlying blob.

- Layer 3. This layer contains only one node. This node, labeled C, performs competitive learning. The inputs to this node are the similarity degrees provided from all the existing blobs. The output of this node, $O^{(3)}$, takes the largest of its inputs, i.e.,

$$O^{(3)} = \max_{1 \leq j \leq J} O_j^{(2)}. \tag{7}$$

Let blob a be the winner blob, i.e.,

$$a = \arg \max_{1 \leq j \leq J} O_j^{(2)}. \tag{8}$$

- Output layer. The layer contains a single node. It applies its input through a hard limit function as follows:

$$O^{(4)} = \text{hardlim}\left(O^{(3)}\right) = \begin{cases} 1, & \text{if } O^{(3)} \geq \rho \\ 0, & \text{otherwise} \end{cases} \tag{9}$$

where ρ is a predefined threshold. A smaller value of ρ may result in larger blobs, and, therefore, as ρ increases, the number of blobs may also increase. Two cases may occur:

Case I. If $O^{(4)} = 0$, there are no existing blobs to which $\mathbf{p}_i$ is similar enough. In this case, we add a new blob by

$$J = J + 1, \tag{10}$$

$$\boldsymbol{\mu}_J = \mathbf{p}_i, \tag{11}$$

$$s_J = 1 \tag{12}$$

and

$$\sigma_{J,k} = \begin{cases} \sigma_0, & \text{if component } k \text{ is continuous} \\ 0, & \text{if component } k \text{ is discrete} \end{cases} \tag{13}$$

for $1 \leq k \leq m$. Note that blob J contains pattern $\mathbf{p}_i$ only, and, obviously, its deviation is 0. Since we cannot use zero deviation in the calculation of the component membership degrees in Equation (4) for continuous components, we set it to be σ_0 which is a user-defined constant. For discrete components, only match or no-match is allowed. Therefore, we set the deviation of discrete components to be zero.

Case II. If $O^{(4)} = 1$, we regard $\mathbf{p}_i$ to be most similar to blob a. Then we assign $\mathbf{p}_i$ to blob a, and $\boldsymbol{\mu}_a$ and $\boldsymbol{\sigma}_a$ of blob a are modified to include $\mathbf{p}_i$ as its member:

$$\mu_{a,k} = \begin{cases} \frac{s_a \times \mu_{a,k} + p_{i,k}}{s_a + 1}, & \text{if component } k \text{ is continuous} \\ \mu_{a,k}, & \text{if component } k \text{ is discrete} \end{cases} \tag{14}$$

$$\sigma_{a,k} = \begin{cases} \sqrt{\frac{(s_a^2 - 1)(\sigma_{a,k} - \sigma_0)^2 + s_a\left(p_{i,k} - \mu_{a,k}\right)^2}{s_a(s_a+1)}} + \sigma_0, & \text{if component } k \text{ is continuous} \\ 0, & \text{if component } k \text{ is discrete} \end{cases} \tag{15}$$

for $1 \leq k \leq m$. Finally, we update the size of blob a:

$$s_a = s_a + 1. \tag{16}$$

Note that J is not changed in this case.

The above process is iterated for all the patterns in S_{tr}. In the end, we have J blobs. Note that the training patterns in a blob have a high degree of similarity to each other. We regard the statistical mean of each blob, $\boldsymbol{\mu}_j$, $1 \le j \le J$, as a prototype representing all the patterns included in the blob. Let

$$S'_{tr} = \{(\mathbf{x}'_1, y'_1), (\mathbf{x}'_2, y'_2), \ldots, (\mathbf{x}'_J, y'_J)\} \tag{17}$$

where

$$\begin{aligned} \mathbf{x}'_j &= [\mu_{j,1}, \mu_{j,2}, \ldots, \mu_{j,m-1}], \\ y'_j &= \mu_{j,m} \end{aligned} \tag{18}$$

for $1 \le j \le J$. We have reduced the training dataset S_{tr} with ℓ patterns to another training dataset S'_{tr} with J patterns. Obviously, $J \le \ell$. Then we may construct a classification or regression model from S'_{tr} instead of directly from S_{tr}. Since S'_{tr} is smaller than S_{tr}, the computation cost of model construction can be less. The proposed prototype reduction method can be summarized in Algorithm 1.

Algorithm 1 Similarity-based prototype reduction algorithm

Input: (1) Original training dataset S_{tr} of Equation (1).
(2) Threshold: ρ.
(3) Initial standard deviation: σ_0.

Initialization: $J = 0$.

1: Blob 1 is created by Equations (10)-(13).
2: **for** $i = 2$ to ℓ **do**
3: Calculate the component membership degree $O^{(1)}_{j,k}$ of $\mathbf{p}_i$ to blob j by Equation (4), $k = 1, 2, \ldots, m$, $j = 1, \ldots, J$.
4: Calculate the similarity degree $O^{(2)}_j$ of $\mathbf{p}_i$ to blob j by Equation (6), $j = 1, \ldots, J$.
5: Find the winner blob, blob a, by Equation (8).
6: Calculate $O^{(4)}$ by Equation (9).
7: **if** $O^{(4)} = 0$ **then**
8: A new blob is created by Equations (10)-(12).
9: **else**
10: Include $\mathbf{p}_i$ in blob a by Equations (14)-(16).
11: **end if**
12: **end for**

Output: Reduced training dataset S'_{tr} of Equation (17).

As for the specification of ρ, the threshold for a discrete component should be 1, dictating a total match for discrete components. A threshold for a continuous component can be any value in $(0, 1]$. For simplicity, the user can specify the same threshold for all continuous components, as shown in the examples later.

4.2. **Complexity comparison.** For SS, J out of the ℓ patterns are sampled, and thus the time complexity involved with SS is $O(J)$. So SS is fast. However, SS does not take into account the characteristics inherently associated with the original training dataset. Next, we estimate the complexity for the k-means algorithm. For convenience, assume that each bin contains p patterns which are to be divided into t clusters and each cluster contains v patterns. Therefore, we have $p = \ell/K$, $t = J/K$ and $v = \ell/J$. In each iteration involved in a bin, t distances have to be computed and $t-1$ comparisons have to be done for each pattern. Also, $v-1$ additions and one division are performed to obtain the new centroid mean for a cluster. Therefore, $[p(t+t-1)+tv]I$ computations are required for a bin, where I indicates the number of iterations to be performed before convergence. Then

the total number of computations required for k-means is $K[p(t+t-1)+tv]I$ which is of order $O(J\ell I/K)$.

For DROP3, it first checks if any patterns are misclassified by their k nearest neighbors. For each pattern, $\ell-1$ distances are computed and its k nearest neighbors are obtained, costing a time proportional to $2\ell + k\log\ell$. So, the time is $\ell(\ell-1+2\ell+k\log(\ell))$ in total for this step. Next, a decreasing list of distances to the nearest enemies are created, costing a time proportional to $\ell\log\ell$. Finally, a pattern is removed if its removal does not deteriorate the classification accuracy induced. This takes a time proportional to ℓT_c where T_c is the time required for classification. Therefore, the total time complexity involved with DROP3 is $\ell(\ell-1+2\ell+k\log\ell)+\ell\log\ell+\ell T_c$ which is of order $O(\ell^2+\ell T_c)$. For RSP3, the distance between any two patterns has to be computed in order to determine the two patterns that are farthest apart. Obviously, the time complexity involved is at least of order $O(\ell^2)$. For HMN-EI, the time complexity is $O(\ell^2)$ [13]. For our algorithm, we only have to process each pattern once. Since there are J blobs to be considered, the time complexity involved in our algorithm is $O(J\ell)$.

5. Examples.

We give two examples here to illustrate how the proposed method works for prototype reduction.

5.1. Example 1.

Let S_{tr} contain 10 patterns $(\mathbf{x}_1, y_1)$, $(\mathbf{x}_2, y_2)$, $\ldots$, $(\mathbf{x}_{10}, y_{10})$ as listed below:

$$\begin{aligned}
\mathbf{x}_1 &= [0.85, 0.88], \quad y_1 = 0.266;\\
\mathbf{x}_2 &= [0.73, 0.86], \quad y_2 = 0.227;\\
\mathbf{x}_3 &= [0.78, 0.83], \quad y_3 = 0.310;\\
\mathbf{x}_4 &= [0.18, 0.19], \quad y_4 = 0.018;\\
\mathbf{x}_5 &= [0.25, 0.12], \quad y_5 = 0.023;\\
\mathbf{x}_6 &= [0.09, 0.21], \quad y_6 = 0.005;\\
\mathbf{x}_7 &= [0.32, 0.37], \quad y_7 = 0.094;\\
\mathbf{x}_8 &= [0.64, 0.72], \quad y_8 = 0.316;\\
\mathbf{x}_9 &= [0.82, 0.77], \quad y_9 = 0.445;\\
\mathbf{x}_{10} &= [0.13, 0.24], \quad y_{10} = 0.012.
\end{aligned}$$

All the components in a pattern are continuous.

We run the proposed algorithm, by setting $\rho = \log(0.8^3) = 3\times\log(0.8) = -0.6694$ and $\sigma_0 = 0.35$, on the patterns of S_{tr}. Note that the same threshold, 0.8, is set for all continuous components. Firstly, we consider the pattern

$$\mathbf{p}_1 = [0.85, 0.88, 0.266], \tag{19}$$

corresponding to $(\mathbf{x}_1, y_1)$, and create the first blob by

$$\begin{aligned}
\boldsymbol{\mu}_1 &= [0.85, 0.88, 0.266],\\
\boldsymbol{\sigma}_1 &= [0.35, 0.35, 0.35]
\end{aligned}$$

and we have $J = 1$. Then, we consider the second pattern $\mathbf{p}_2$, corresponding to $(\mathbf{x}_2, y_2)$. The operations performed are shown below.

- Layer 1. The component membership degrees are

$$O_{1,1}^{(1)} = 0.889, \quad O_{1,2}^{(1)} = 0.997, \quad O_{1,3}^{(1)} = 0.988.$$

- Layer 2. The similarity degree is

$$O_1^{(2)} = \log(0.889) + \log(0.997) + \log(0.988) = -0.133.$$

- Layer 3. The winner blob is blob 1 and

$$O^{(3)} = -0.133 > \rho.$$

- Output layer. Since $O^{(3)} > \rho$, $\mathbf{p}_2$ is assigned to blob 1, and $\boldsymbol{\mu}_1$ and $\boldsymbol{\sigma}_1$ are modified to become

$$\boldsymbol{\mu}_1 = [0.790, 0.870, 0.247],$$
$$\boldsymbol{\sigma}_1 = [0.435, 0.364, 0.378].$$

When all the training patterns in S_{tr} have been considered, we obtain 2 blobs as shown in Table 1. The reduced dataset S'_{tr} then contains the following two patterns:

TABLE 1. Two blobs obtained for Example 1

blob	size s_j	mean $\boldsymbol{\mu}_j$	standard deviation $\boldsymbol{\sigma}_j$
$j = 1$	5	$[0.764, 0.812, 0.313]$	$[0.433, 0.416, 0.432]$
$j = 2$	5	$[0.194, 0.226, 0.031]$	$[0.443, 0.442, 0.386]$

$$\mathbf{x}'_1 = [0.764, 0.812], \quad y'_1 = 0.313;$$
$$\mathbf{x}'_2 = [0.194, 0.226], \quad y'_2 = 0.031.$$

That is, $S'_{tr} = \{(\mathbf{x}'_1, y'_1), (\mathbf{x}'_2, y'_2)\}$. Note that S_{tr} contains 10 patterns, while S'_{tr} contains only 2 patterns. Then we can build a regression model from S'_{tr} instead of directly from S_{tr}. Since S'_{tr} is smaller than S_{tr}, the modeling process involved with S'_{tr} requires less memory and time than that involved directly with S_{tr}.

5.2. **Example 2.** Let S_{tr} contain 10 patterns $(\mathbf{x}_1, y_1)$, $(\mathbf{x}_2, y_2)$, ..., $(\mathbf{x}_{10}, y_{10})$ of two classes, as listed below:

$$\mathbf{x}_1 = [0.82, 0.69, 0.73, 0.92, 0.58, a, d, h], \quad y_1 = 1;$$
$$\mathbf{x}_2 = [0.53, 0.49, 0.18, 0.37, 0.89, a, f, k], \quad y_2 = 2;$$
$$\mathbf{x}_3 = [0.78, 0.82, 0.75, 0.88, 0.63, a, d, h], \quad y_3 = 1;$$
$$\mathbf{x}_4 = [0.81, 0.68, 0.76, 0.87, 0.62, a, d, h], \quad y_4 = 1;$$
$$\mathbf{x}_5 = [0.52, 0.47, 0.25, 0.33, 0.65, a, f, k], \quad y_5 = 2;$$
$$\mathbf{x}_6 = [0.51, 0.32, 0.33, 0.49, 0.29, b, e, g], \quad y_6 = 1;$$
$$\mathbf{x}_7 = [0.50, 0.45, 0.22, 0.35, 0.62, a, f, k], \quad y_7 = 2;$$
$$\mathbf{x}_8 = [0.48, 0.39, 0.31, 0.51, 0.25, b, e, g], \quad y_8 = 1;$$
$$\mathbf{x}_9 = [0.25, 0.31, 0.41, 0.19, 0.28, c, e, g], \quad y_9 = 2;$$
$$\mathbf{x}_{10} = [0.27, 0.30, 0.40, 0.20, 0.30, c, e, g], \quad y_{10} = 2.$$

The first five components of a pattern are continuous and the remaining ones are discrete.

We run the proposed algorithm with $\rho = \log(0.8^5 \times 1^4) = 5 \times \log(0.8) = -1.116$ and $\sigma_0 = 0.25$ on S_{tr}. Note that the same threshold, 0.8, is set for all continuous components, and the threshold 1 is set for all discrete components. The first blob created from $\mathbf{p}_1 = (\mathbf{x}_1, y_1)$ has the mean

$$\boldsymbol{\mu}_1 = [0.82, 0.69, 0.73, 0.92, 0.58, a, d, h, 1]$$

and deviation

$$\boldsymbol{\sigma}_1 = [0.25, 0.25, 0.25, 0.25, 0.25, 0, 0, 0, 0].$$

The operations performed for the second pattern, $\mathbf{p}_2 = (\mathbf{x}_2, y_2)$, are:

- Layer 1. The component membership degree vector is

$$\mathbf{O}_1^{(1)} = [0.260, 0.527, 0.008, 0.007, 0.215, 1, 0, 0, 0].$$

- Layer 2. The similarity degree is

$$\begin{aligned} O_1^{(2)} &= \log(0.260) + \log(0.527) + \log(0.008) + \log(0.007) \\ &\quad + \log(0.215) + \log(1) + \log(0) + \log(0) + \log(0)) \\ &= -\infty. \end{aligned}$$

- Layer 3. The winner blob is blob 1 and

$$O^{(3)} = -\infty < \rho.$$

- Output layer. Since $O^{(3)} < \rho$, we have $O^{(4)} = 0$. Thus, a new blob is created for $\mathbf{p}_2$:

$$\begin{aligned} \boldsymbol{\mu}_2 &= [0.53, 0.49, 0.18, 0.37, 0.89, a, f, k, 2], \\ \boldsymbol{\sigma}_2 &= [0.25, 0.25, 0.25, 0.25, 0.25, 0, 0, 0, 0]. \end{aligned}$$

Then, we consider the third pattern, $\mathbf{p}_3 = (\mathbf{x}_3, y_3)$, giving:

- Layer 1. The component membership degree vector is

$$\begin{aligned} \mathbf{O}_1^{(1)} &= [0.975, 0.986, 0.994, 0.961, 0.961, 1, 1, 1, 1], \\ \mathbf{O}_2^{(1)} &= [0.368, 0.429, 0.006, 0.016, 0.339, 1, 0, 0, 0]. \end{aligned}$$

- Layer 2. The similarity degrees are

$$\begin{aligned} O_1^{(2)} &= \log(0.975) + \log(0.986) + \log(0.994) + \log(0.961) \\ &\quad + \log(0.961) + \log(1) + \log(1) + \log(1) + \log(1) \\ &= -0.1264, \\ O_2^{(2)} &= \log(0.368) + \log(0.429) + \log(0.006) + \log(0.016) \\ &\quad + \log(0.339) + \log(1) + \log(0) + \log(0) + \log(0) \\ &= -\infty. \end{aligned}$$

- Layer 3. The winner blob is blob 1 and

$$O^{(3)} = -0.1264 > \rho.$$

- Output layer. Since $O^{(3)} > \rho$, we have $O^{(4)} = 1$. Thus, $\mathbf{p}_3$ is assigned to blob 1, and $\boldsymbol{\mu}_1$ and $\boldsymbol{\sigma}_1$ are modified to become

$$\begin{aligned} \boldsymbol{\mu}_1 &= [0.800, 0.705, 0.740, 0.905, 0.605, a, d, h, 1], \\ \boldsymbol{\sigma}_1 &= [0.278, 0.271, 0.264, 0.285, 0.285, 0, 0, 0, 0]. \end{aligned}$$

When all the training patterns in S_{tr} have been considered, we obtain 4 blobs, i.e., $J = 4$, and the reduced dataset S'_{tr} contains the following four patterns:

$$\begin{aligned} \mathbf{x}'_1 &= [0.803, 0.697, 0.747, 0.893, 0.610, a, d, h], \quad y'_1 = 1; \\ \mathbf{x}'_2 &= [0.517, 0.470, 0.217, 0.350, 0.720, a, f, k], \quad y'_2 = 2; \\ \mathbf{x}'_3 &= [0.495, 0.355, 0.320, 0.500, 0.270, b, e, g], \quad y'_3 = 1; \\ \mathbf{x}'_4 &= [0.260, 0.305, 0.405, 0.195, 0.290, c, e, g], \quad y'_4 = 2. \end{aligned}$$

That is, $S'_{tr} = \{(\mathbf{x}'_1, y'_1), (\mathbf{x}'_2, y'_2)\}$, $S'_{tr} = \{(\mathbf{x}'_1, y'_1), (\mathbf{x}'_2, y'_2), (\mathbf{x}'_3, y'_3), (\mathbf{x}'_4, y'_4)\}$. Note that S_{tr} contains 10 patterns, while S'_{tr} contains only 4 patterns. Then we can build a classification model from S'_{tr} instead of directly from S_{tr}. Since S'_{tr} is smaller than S_{tr}, the modeling process involved with S'_{tr} requires less memory and time than that involved directly with S_{tr}.

6. **Experimental Results.** In this section, we present some experimental results to show the effectiveness of our proposed prototype reduction method. We also compare it with three instance-filtering based methods, SS [18,19], DROP3 [9], HMN-EI [13], and two instance-abstraction based methods, k-means [26-28] and RSP3 [10], using several classification and regression datasets. As described in Section 3, DROP3, HMN-EI and RSP3 can only be applied to classfication problems. Support vector machines (SVM) [14,15,29] and support vector regression (SVR) [29] are adopted to construct classification and regression models, respectively, from given training datasets. We use a computer with Intel(R) Core(TM)2 Quad CPU Q6600 2.40GHz and 4GB of RAM to conduct the experiments. The software used is C#.

For convenience, our proposed similarity-based prototype reduction method is abbreviated as SBPR in the following presentation. Note that the order in which the training patterns are considered influences the blobs obtained. We use a heuristic for the entry of the training patterns. We sort all the patterns, in decreasing order, by their largest continuous components, and enter the patterns in this order. In this way, more significant patterns will be entered first and likely become the core of the underlying blob. For example, let $\mathbf{p}_1 = [0.1, 0.3, 0.6]$, $\mathbf{p}_2 = [0.3, 0.3, 0.4]$ and $\mathbf{p}_3 = [0.8, 0.1, 0.1]$ be three patterns. The largest components in these patterns are 0.6, 0.4 and 0.8, respectively. The sorted list is 0.8, 0.6, 0.4. So the order of entry is $\mathbf{p}_3$, $\mathbf{p}_1$ and $\mathbf{p}_2$. We will show later the effect of other possible orderings.

6.1. **Experiment I.** In this experiment, we compare different reduction methods on three classification datasets. The datasets used are letter and shuttle from the Statlog collections [30], and the 1999 KDD Cup from the 1998 DARPA Intrusion Detection Evaluation Program administered by the MIT Lincoln Lab [31]. The letter dataset contains 20,000 unique letter images composed of the uppercase letters A to Z in 20 different fonts. Each image was distorted both horizontally and vertically but still remained recognizable to humans. Each image was converted into 16 primitive numerical features. We use 15,000 patterns for training and the rest for testing. The shuttle dataset contains 58,000 patterns composed of 7 classes. Each pattern consists of 9 numerical features. This dataset has a skewed distribution and approximately 80% of the patterns belong to class 1. We use 43,500 patterns for training and the rest for testing. In the 1999 KDD dataset, 4,898,431 connection records are used for training and 311,029 connection records are used for testing. Both training and testing datasets 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 pattern consists of 34 continuous components and 7 discrete components. This dataset has a skewed distribution, approximately 20% of the patterns belonging to class Normal and approximately 80% of the patterns belonging to class DoS.

Note that SS and k-means require the number of representatives to be set in advance. To meet this requirement, we ran SBPR first and then set the number of representatives for SS and k-means to be that obtained by SBPR. DROP3, HMN-EI and RSP3 do not require the number of representatives to be specified in advance. Since SS and k-means are sensitive to random selection, we repeated 10 times for each dataset and then averaged

Table 2. Comparison on letter for Experiment I

	WO-red	SBPR	SS	DROP3	HMN-EI	k-means	RSP3
#-rep	15000	4371	4371	5097	11573	4371	5821
R-rate (%)	0.00	70.86	70.86	66.02	22.85	70.86	61.19
PR-time (sec)	0.000	5.313	0.018	498.520	637.751	97.473	152.412
MG-time (sec)	81.141	18.719	17.453	20.281	55.859	17.953	25.500
C-acc (%)	97.98	97.06	94.07	95.36	95.26	96.71	97.66

Table 3. Comparison on shuttle for Experiment I

	WO-red	SBPR	SS	DROP3	HMN-EI	k-means	RSP3
#-rep	43500	280	280	15466	28435	280	279
R-rate (%)	0.00	99.36	99.36	64.45	34.63	99.36	99.36
PR-time (sec)	0.000	6.334	0.014	3301.833	3359.776	37.517	1477.325
MG-time (sec)	137.297	0.719	0.797	38.781	88.172	0.688	0.797
C-acc (%)	99.79	99.72	96.68	99.86	99.90	99.32	98.54

the results. The results for the letter dataset are summarized in Table 2 in which "#-rep" stands for the number of representatives obtained, "R-rate" for reduction rate, "PR-time" for prototype reduction time, "MG-time" for model generation time by SVM, and "C-acc" for classification accuracy of the obtained model. The column marked with "WO-red" in the table indicates that the results are obtained without prototype reduction. The reduction rate is the ratio $(\ell - J)/\ell$ where ℓ is the number of original patterns and J is the number of representatives obtained. For letter, the number of original patterns is 15,000, so the number of representatives indicated in the "WO-red" column is 15,000. The reduction rate for letter by SBPR is $(15000 - 4371)/15000 = 70.86\%$. The accuracy for letter achieved by SBPR is 97.06% which is better than 94.07% by SS, 95.36% by DROP3, and 95.26% by HMN-EI. SBPR gets higher accuracy than k-means, i.e., 97.06% vs. 96.71%. RSP3 gets higher accuracy than SBPR, i.e., 97.66% vs. 97.06%, but SBPR runs over 25 times faster than RSP3. Note that SBPR spent 5.313 seconds in prototype reduction and SVM spent 18.719 seconds in generating a classification model from the prototypes obtained by SBPR. So the total time by SBPR is $5.313 + 18.719 = 24.032$ seconds which is lower than the 81.141 seconds by WO-red. This indicates that SBPR is effective in reducing the computation time for classification in this case. However, the prototype reduction time taken by any of DROP3, HMN-EI, k-means, and RSP3 is larger than the model generation time without reduction. The application of these methods increases, rather than decreases, the computation time for classification in this case.

The results for the shuttle dataset are summarized in Table 3. For shuttle, SBPR, DROP3 and HMN-EI have comparable accuracies, but are more accurate than SS. SBPR runs much faster than DROP3 and HMN-EI. SBPR takes 6.334 seconds, while DROP3 and HMN-EI take 3301.833 and 3359.776 seconds, respectively. SBPR and k-means get equally good accuracies. However, SBPR runs over 6 times faster than k-means. SBPR gets a higher accuracy than RSP3, i.e., 99.72% vs. 98.54%. Furthermore, SBPR runs over 230 times faster than RSP3. Note that SBPR spent 6.334 seconds in prototype reduction and SVM spent 0.719 seconds in generating a classification model from the prototypes obtained by SBPR. So the total time by SBPR is $6.334 + 0.719 = 7.053$ seconds which is much lower than the 137.297 seconds by WO-red. This indicates that SBPR is effective in reducing the computation time for classification in this case. However, the prototype

TABLE 4. Comparison on 1999 KDD Cup for Experiment I

	WO-red	SBPR	SS	DROP3	HMN-EI	k-means	RSP3
#-rep	4898431	3126	3126	—	—	—	—
R-rate (%)	—	99.94	99.94	—	—	—	—
PR-time (sec)	—	1595.921	0.925	—	—	—	—
MG-time (sec)	—	13.828	10.295	—	—	—	—
C-acc (%)	—	92.55	86.19	—	—	—	—

TABLE 5. Comparison on 10% 1999 KDD Cup for Experiment I

	WO-red	SBPR	SS	DROP3	HMN-EI	k-means	RSP3
#-rep	494021	1515	1515	—	—	1515	—
R-rate (%)	0.00	99.69	99.69	—	—	99.69	—
PR-time (sec)	0.000	443.168	0.083	—	—	5214.800	—
MG-time (sec)	4478.359	4.843	4.109	—	—	4.672	—
C-acc (%)	92.51	92.04	81.05	—	—	92.09	—

reduction time taken by any of DROP3, HMN-EI, and RSP3 is larger than the model generation time without reduction.

The results for the 1999 KDD Cup dataset are summarized in Table 4. Since the 1999 KDD Cup dataset contains about 5 million training patterns, all the methods, except SBPR and SS, have troubles in loading the whole dataset into the memory and get a "out of memory" error. Therefore, they cannot be applied to this dataset, as indicated by "—" in this table. In practice, prototype reduction is required before a classification model can be built for such a huge training dataset. DROP3, HMN-EI, k-means and RSP3 demand a huge amount of memory and are not applicable for this case. SBPR and SS are applicable, and SS runs much faster than SBPR. However, the accuracy obtained by SS is much lower than that obtained by SBPR. SBPR gets 92.55% in accuracy, while SS gets only 86.19% in accuracy. To avoid the "out of memory" error, we use 10% of the original training patterns, as that commonly done in other research works, and run the experiment again. The results for this 10% 1999 KDD Cup dataset are shown in Table 5. Note that RSP3, DROP3 and HMN-EI still have troubles in this case. They ran over 100,000 seconds without termination. The corresponding entries are marked with "—". Apparently, SS runs much faster but has much lower accuracy than the other methods. SBPR and k-means get equally good accuracies. However, SBPR runs much faster than k-means. SBPR takes 443.168 seconds while k-means requires 5214.800 seconds. Besides, the prototype reduction time taken by k-means is larger than the model generation time without reduction.

6.2. **Experiment II.** In this experiment, we compare different reduction methods on two regression datasets. The datasets used are houses from the StatLib datasets archive [32] and comp-activ from the Delve datasets archive [33]. The houses dataset contains 20,640 patterns for predicting the price of houses in California. The input variables include median income, housing median age, total rooms, total bedrooms, population, households, latitude and longitude. The output variable is the median house value. The comp-activ dataset contains 8,192 patterns for predicting a computer system activity in a multi-processor, multi-user computer system. As that commonly done in research, we select the "cpuSmall" subset which contains 12 input variables. The output variable is the proportion of the CPU time spent in the user mode. For each dataset, threefold

TABLE 6. Comparison for Experiment II

	houses				comp-activ			
	WO-red	SBPR	k-means	SS	WO-red	SBPR	k-means	SS
#-rep	13760	162	162	162	5461	247	247	247
R-rate (%)	0.00	98.83	98.83	98.83	0.00	95.48	95.48	95.48
PR-time (sec)	0.000	1.796	20.762	0.006	0.000	0.914	38.717	0.002
MG-time (sec)	1136.651	0.792	0.635	0.589	346.865	2.089	1.927	1.323
MSE	0.0168	0.0212	0.0219	0.0249	0.0039	0.0054	0.0062	0.0081

TABLE 7. Results by SBPR with different settings of ρ for classification

	letter		shuttle		1999 KDD Cup	
	$t = 0.80$	$t = 0.90$	$t = 0.80$	$t = 0.90$	$t = 0.80$	$t = 0.90$
#-rep	4371	10108	280	463	1515	2703
R-rate (%)	70.06	32.61	99.36	98.94	99.69	99.45
PR-time (sec)	5.313	20.624	6.334	7.488	443.168	513.745
MG-time (sec)	18.719	51.500	0.719	1.219	4.843	9.219
C-acc (%)	97.06	97.96	99.72	99.86	92.04	91.88

cross-validation is applied, i.e., two-thirds of the patterns are used for training and the remaining one-third for testing.

We compare SBPR with SS and k-means, since RSP3, DROP3 and HMN-EI are not applicable to regression datasets. The results are summarized in Table 6 in which the number of representatives, reduction rate, prototype reduction time, model generation time by SVR, and mean square error (MSE) are listed. We repeated 10 times for each dataset and then averaged the results for SS and k-means. Note that SS runs much faster than the other methods. However, it selects inappropriate patterns to be representatives which result in lower accuracy than the other methods. SBPR performs slightly better than k-means in MSE. Also, SBPR runs much faster than k-means. SBPR takes 1.796/0.914 seconds while k-means requires 20.762/38.717 seconds for houses/comp-activ. Note that SBPR spent 1.796 seconds in prototype reduction and SVR spent 0.792 seconds in generating a regression model from the prototypes obtained by SBPR for the houses dataset. So the total time by SBPR is $1.796 + 0.792 = 2.588$ seconds which is much lower than the 1136.651 seconds by WO-red. Furthermore, the total time taken by either k-means or SS is much less than the model generation time without reduction. This indicates that any of SBPR, k-means, and SS is effective in reducing the time requirement for regression in this case.

6.3. **Experiment III.** We investigate the effect of different settings for SBPR. As mentioned earlier, the setting of ρ may affect the number of blobs SBPR generates. As ρ increases, the patterns in a blob are required to be more similar to each other and thus the number of blobs obtained also increases. However, the performance of the obtained representatives does not vary significantly as ρ changes. Table 7 shows the results obtained by SBPR with different settings of ρ for the three classification datasets used in Experiment I. In this table, ρ is set to be $D \times \log(t)$ where D is the number of continuous components, i.e., $D = 16$ for letter, $D = 9$ for shuttle, and $D = 34$ for 1999 KDD Cup. By changing t we can have different settings for ρ and thus obtain different numbers of representatives. However, the resulting accuracies do not differ significantly. For example, SBPR gets 4,371 representatives and 97.06% in accuracy with $t = 0.80$, and 10,108 representatives and 97.96% in accuracy with $t = 0.90$ for the letter dataset. For 1999

TABLE 8. Results by SBPR with different settings of ρ for regression

	houses			comp-activ		
	$t = 0.70$	$t = 0.80$	$t = 0.90$	$t = 0.70$	$t = 0.80$	$t = 0.90$
#-rep	73	162	674	114	247	732
R-rate (%)	99.47	98.83	95.10	97.92	95.48	86.60
PR-time (sec)	1.610	1.866	3.345	0.697	0.754	1.141
MG-time (sec)	0.286	0.828	5.333	0.667	2.208	9.589
MSE	0.0284	0.0212	0.0187	0.0080	0.0054	0.0046

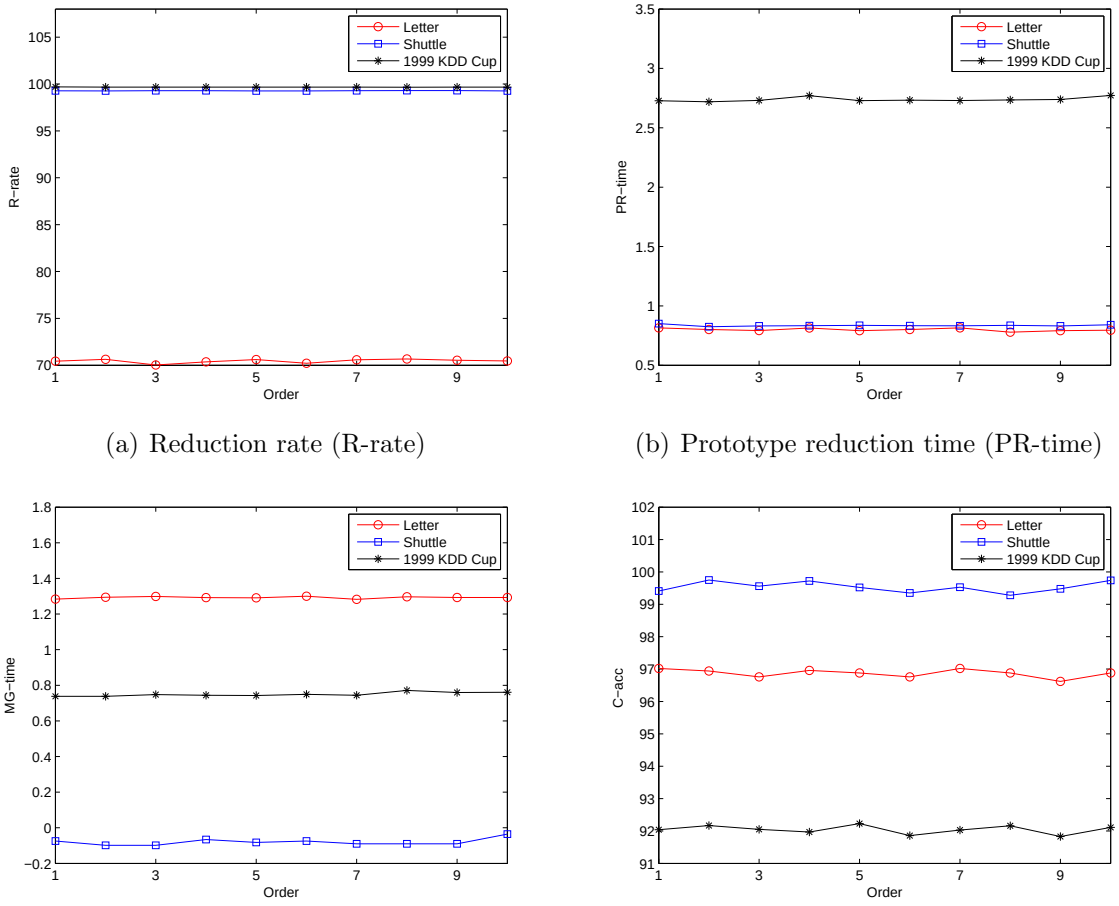
KDD Cup, SBPR gets 1,515 representatives with $t = 0.80$ and 2,703 representatives with $t = 0.90$, but the accuracies for both cases are almost the same.

Table 8 shows the results obtained by SBPR with different settings of ρ for the two regression datasets used in Experiment II. Note that $D = 9$ for houses and $D = 13$ for comp-activ. Again, by changing t we can have different settings of ρ and thus obtain different numbers of representatives. However, the MSE values obtained do not differ significantly. For example, SBPR gets 162 representatives and 0.0212 in MSE with $t = 0.80$, and 674 representatives and 0.0187 in MSE with $t = 0.90$ for the houses dataset. For comp-activ, SBPR gets 247 representatives with $t = 0.80$ and 732 representatives with $t = 0.90$, but the MSE values only differ slightly.

The order in which the patterns are considered in SBPR may also affect the number of blobs generated. However, the performance of the obtained representatives does not vary significantly. To investigate the influence imposed by the order, we arrange the presentation of training patterns to SBPR in ten different sequences, labeled by seq#1 to seq#10. Figure 2 shows the results for the three classification datasets used in Experiment I. The ten different sequences are indicated horizontally in the figure. From this figure, we can see that the classification accuracy is not sensitive to the presentation order in which training patterns are considered in SBPR.

7. **Conclusion.** We have presented a similarity-based prototype reduction (SBPR) method to reduce the number of training patterns for supervised learning. Training patterns are considered one at a time and those similar to each other are grouped into the same blob. Each blob is characterized by a membership function with a statistical mean and standard deviation. When a pattern is assigned to an existing blob, the membership function of that blob is updated. If a pattern is not similar to any existing blob, a new blob is created. When all the patterns have been considered, a desired number of blobs are formed automatically. Then the statistical mean of each blob is regarded as a prototype representing all the patterns included in the blob. The distribution of the data contained in each blob is statistically well described. Each obtained prototype is a good representative of the corresponding blob. Blobs are generated automatically based on the similarity relationship among the original training patterns. The proposed method can be applied efficiently to both classification and regression problems.

SBPR, unlike k-means, goes through each training pattern only once. Therefore, it is efficient and can work on very large training datasets. Different numbers of representatives are extracted according to the distribution of the original training patterns. For those patterns with a dense distribution, SBPR extracts a small number of representatives. On the contrary, SBPR extracts more representatives out from the patterns with a sparse distribution. For example, class Normal in the 10% 1999 KDD Cup dataset has a sparse distribution, while class DoS has a dense distribution. Among the 494,021 original training patterns, 97,278 ones belong to class Normal and 391,458 ones belong to class DoS.



(a) Reduction rate (R-rate)

(b) Prototype reduction time (PR-time)

(c) Model generation time (MG-time)

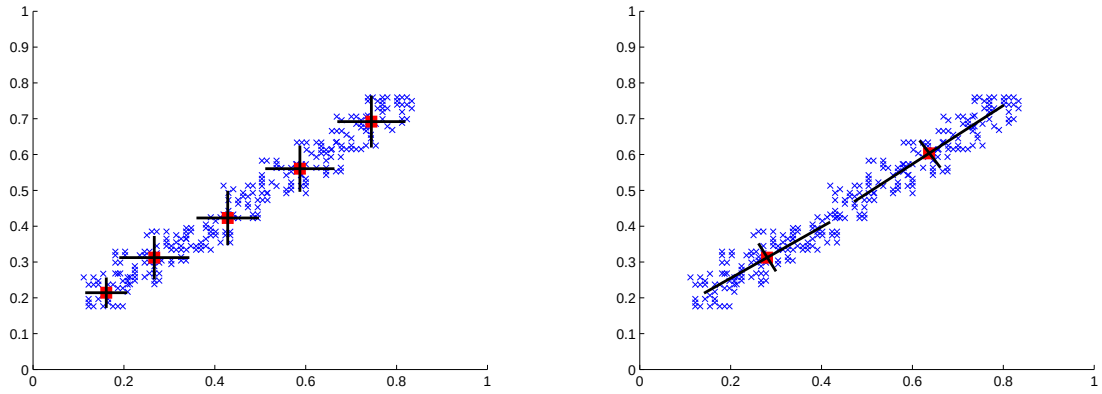
(d) Classification accuracy (C-acc)

FIGURE 2. Results by SBPR with different presentation orders for classification

However, SBPR extracts 938 representatives for class Normal in Experiment I. On the contrary, SBPR extracts 324 representatives for class DoS.

The blobs obtained by SBPR cannot be slanted ones. A slanted blob may sometimes fit a set of patterns better than a non-slanted one. We are investigating the possibility of incorporating principal component analysis (PCA) [34] with SBPR to generate slanted blobs. The original training patterns are transformed to another space and the similarities are calculated in that space. In this way, the locations, orientation, and the number of blobs obtained can reflect more truthfully the characteristics of the original training dataset. For example, Figure 3 shows the results of applying SBPR without and with PCA, respectively, on a synthetic dataset which contains 270 two-dimensional patterns. By SBPR only, with $\rho = \log(0.5^2)$ and $\sigma_0 = 0.08$, 5 blobs are generated, each marked with a cross, as shown in Figure 3(a). Therefore, five representatives are obtained. On the contrary, by incorporating PCA with SBPR, only 2 blobs are generated, as shown in Figure 3(b), and thus two representatives are obtained for the same original dataset. Note that the distribution of the data can be better described by the oblique hyper-ellipsoidal blobs with the help of PCA. However, incorporating PCA with SBPR requires extra time. How to incorporate PCA with SBPR to generate slanted blobs efficiently may become a major computation issue.

We have applied SBPR to reducing the dimensionality of the features involved in supervised learning [35]. In some areas, e.g., text processing, the dimensionality of the



(a) Blobs obtained by SBPR (b) Blobs obtained by SBPR with PCA

FIGURE 3. Blobs obtained by SBPR without and with PCA

feature vector is usually very large. Feature reduction algorithms are highly demanded for dealing with high-dimensional document datasets efficiently. We are investigating the possibility of incorporating feature reduction and prototype reduction together to further reduce the train set size for machine learning. We are also interested in applying SBPR to other problems, such as image segmentation, fuzzy modeling and web mining.

Acknowledgment. This work was partially supported by "Aim for the Top University Plan" of the National Sun Yat-sen University and Ministry of Education, and by the National Science Council under the grants NSC-97-2221-E-110-048-MY3 and NSC-98-2221-E-110-052. The authors are grateful to the anonymous reviewers for their comments, which were very helpful in improving the quality and presentation of the paper.

REFERENCES

[1] M. F. Hassan and K. Boukas, Multilevel technique for large scale lqr with time-delays and systems constraints, *International Journal of Innovative Computing, Information and Control*, vol.3, no.2, pp.419-434, 2007.

[2] S. Tong, W. Wang and L. Qu, Decentralized robust control for uncertain t-s fuzzy large-scale systems with time-delay, *International Journal of Innovative Computing, Information and Control*, vol.3, no.3, pp.657-672, 2007.

[3] L. Luoh, Control design of T-S fuzzy large-scale systems, *International Journal of Innovative Computing, Information and Control*, vol.5, no.9, pp.2869-2880, 2009.

[4] G. Feng, G.-B. Huang, Q. Lin and R. Gay, Error minimized extreme learning machine with growth of hidden nodes and incremental learning, *IEEE Transactions on Neural Networks*, vol.20, no.8, pp.1352-1357, 2009.

[5] 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, 2002.

[6] 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, 1972.

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

[8] 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, 1975.

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

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

[11] 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, 2004.
[12] 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, 2006.
[13] E. Marchiori, Hit miss networks with applications to instance selection, *Journal of Machine Learning Research*, vol.9, pp.997-1017, 2008.
[14] C. Cortes and V. Vapnik, Support-vector networks, *Machine Learning*, vol.20, no.3, pp.273-297, 1995.
[15] V. Vapnik, *The Nature of Statistical Learning Theory*, 2nd Edition, Springer, New York, NY, USA, 1999.
[16] 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, 2003.
[17] J. G. Wang, P. Neskovic and L. N. Cooper, Training data selection for support vector machines, *Proc. of the 1st International Conference on Advances in Natural Computation*, pp.554-564, 2005.
[18] E. Pękalska, R. P. W. Duin and P. Paclík, Prototype selection for dissimilarity-based classifiers, *Pattern Recognition*, vol.39, no.2, pp.189-208, 2006.
[19] 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, 2007.
[20] H. Brighton and C. Mellish, On the consistency of information filters for lazy learning algorithms, *Proc. of the 3rd European Conference on Principles of Data Mining and Knowledge Discovery*, pp.283-288, 1999.
[21] P. Datta and D. Kibler, Symbolic nearest mean classifiers, *Proc. of the 14th International Conference on Machine Learning*, pp.82-87, 1997.
[22] J. Macqueen, Some methods for classification and analysis of multivariate observations, *Proc. of the 5th Berkeley Symposium on Mathematical Statistics and Probability*, vol.1, pp.281-297, 1967.
[23] Y. Linde, A. Buzo and R. Gray, An algorithm for vector quantizer design, *IEEE Transactions on Communications*, vol.28, no.1, pp.84-95, 1980.
[24] K. Tabata, M. Sato and M. Kudo, Data compression by volume prototypes for streaming data, *Pattern Recognition*, vol.43, no.9, pp.3162-3176, 2010.
[25] G. J. Klir and B. Yuan, *Fuzzy Sets and Fuzzy Logic: Theory and Applications*, 1st Edition, Prentice Hall PTR, Upper Saddle River, NJ, USA, 1995.
[26] 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, *Proc. of the 6th Brazilian Symposium on Neural Networks*, pp.162-167, 2000.
[27] S. Zheng, X. Lu, N. Zheng and W. Xu, Unsupervised clustering based reduced support vector machines, *Proc. of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp.821-824, 2003.
[28] 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, 2004.
[29] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*, Cambridge University Press, New York, NY, USA, 2000.
[30] D. Michie, D. J. Spiegelhalter and C. C. Taylor, *Machine Learning, Neural and Statistical Classification*, Prentice Hall, Englewood Cliffs, N.J., 1994.
[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, *Proc. of the 2000 DARPA Information Survivability Conference and Exposition*, pp.130-144, 2000.
[32] *Statlib-datasets Archive*, http://lib.stat.cmu.edu/datasets/.
[33] *Delve-datasets*, http://www.cs.toronto.edu/∼mlearn/delve/data/datasets.html.
[34] K. Pearson, On lines and planes of closest fit to systems of points in space, *Philosophical Magazine*, vol.2, no.6, pp.559-572, 1901.
[35] J.-Y. Jiang, R.-J. Liou and S.-J. Lee, A fuzzy self-constructing feature clustering algorithm for text classification, *IEEE Transactions on Knowledge and Data Engineering*, vol.23, no.3, pp.335-349, 2011.