Bootstrapping for efficient handwritten digit recognition

V. Vijaya Saradhi, M. Narasimha Murty*

Department of Computer Science and Automation, Indian Institute of Science, Bangalore 560 012, India

Received 29 July 1999; received in revised form 14 February 2000; accepted 14 February 2000

Abstract

In this paper we present two algorithms for selecting prototypes from the given training data set. Here, we employ the bootstrap technique to preprocess the data. We compare the proposed algorithms with the condensed nearest-neighbor algorithm which is order dependent and a genetic-algorithm-based prototype selection scheme which is order independent. Algorithms proposed in this paper are found to be better than the condensed nearest neighbor and prototype selection methods in terms of classification accuracy. © 2001 Pattern Recognition Society. Published by Elsevier Science Ltd. All rights reserved.

Keywords: Bootstrapping; Redundancy removal; Condensed nearest neighbor; Prototype selection; Genetic algorithms; Thresholding; Classification accuracy

1. Introduction

Pattern classification is an important activity with several applications. In this paper, we consider the application of recognizing handwritten digits. A variety of classifiers [1] are reported in literature. In this work, we use the neighborhood classifiers (NCs) as they are simple. Also, it was reported in [2] that one of the best NCs, the \( k \)-nearest-neighbor classifier (KNNC), is the most robust classifier, among the statistical pattern classifiers, for character recognition. In a practical classification context, a finite collection of labeled patterns, called the training data set, is provided. A neighborhood classifier labels a test pattern [1] based on the labels of its neighboring patterns in the training data set. So, the classification time required to label each of the test patterns increases with the size of the training data set. In most of the real-world applications, the size of the available training data is very large. As a consequence, the time required to classify a test pattern could be prohibitively large. This pragmatic constraint calls for data reduction, that is selecting an appropriate subset of the original training data set without reducing the classification accuracy significantly. This activity of selecting patterns from the training data is also called as prototype selection or exemplar selection [3] and the patterns selected are called prototypes or exemplars.

For data reduction, several methods were proposed in the literature. These include condensed nearest neighbor (CNN) [4] and exemplar selection based on genetic algorithms [3]. Some of these methods are order-dependent. That means the prototypes selected depend on the order of the presentation of the input patterns. CNN falls under the category of order-dependent algorithms, whereas exemplar selection algorithm [3] is order independent. In the current work, we present two methods based on thresholding for selecting prototypes. These prototypes are selected from the bootstrapped data set.

Bootstrapping is a method of mapping or re-sampling the given data. In statistics, it has been used for re-sampling. This method has been used extensively for non-parametric error estimation in the literature [5]. Jain et al. [6] reported results using bootstrapping in estimating the error rate of the nearest-neighbor classifier (NNC) and a quadratic classifier, and have shown experimentally that the error obtained using bootstrapping is
smaller than that using the conventional ‘leave-one-out’ strategy. Chernick et al. [7] compared the performance of a non-parametric estimation scheme with that of the bootstrap technique based on the expected error. Hamamoto et al. [8] developed a bootstrap technique for the NNC to increase its classification accuracy. They generated the bootstrap samples by combining the training data locally. The results reported by them show that the performance of NNC based on bootstrap patterns is superior to that of the KNNC based on the original training data set.

The set of bootstrap samples generated using the algorithm given by Hamamoto et al. [8] might have some redundancies. Here, by redundancy we mean either the same pattern is repeated more than once or some groups of patterns are very similar. Redundancy present in the training data may not help in increasing the classification accuracy. Instead, by retaining only the non-redundant patterns, the performance of NNC based on bootstrap patterns is suited for patterns of high dimensionality. Hence, we use this algorithm in the present study on a large collection of character patterns of high dimensionality. The algorithm is given below.

3. Bootstrap algorithm

Hamamoto et al. have described four algorithms for bootstrapping and reported that one of them is ideally suited for patterns of high dimensionality. Hence, we use this algorithm in the present study on a large collection of character patterns of high dimensionality. The algorithm is given below.

1. $\mathcal{X}^b = \emptyset$.
2. Select a training pattern $\mathbf{X}_i$ from the training data set $\mathcal{X}_a \subset \mathcal{X}$, where $\omega_a$ is the class label of all the patterns in $\mathcal{X}_a$.
3. Find the first $k$ nearest neighbors, $\mathbf{X}_{i1}, \mathbf{X}_{i2}, \ldots, \mathbf{X}_{ik}$ of $\mathbf{X}_i$ in $\mathcal{X}_a$. Here, we assume that $\mathbf{X}_{i1} = \mathbf{X}_a$.
4. Compute the bootstrap sample $\mathbf{X}_a^b$ corresponding to the pattern $\mathbf{X}_i$ as given below

$$\mathbf{X}_i^b = \frac{1}{k} \sum_{j=1}^{k} \mathbf{X}_{ij}$$

5. $\mathcal{X}^b = \mathcal{X}^b \cup \{\mathbf{X}_i^b\}$.
6. Repeat steps 2–5 $\forall a = 1, 2, \ldots, m$ using the sets $\mathcal{X}_a$ comprising patterns in $\mathcal{X}$.

3.1. Why bootstrapping?

A closer examination of the bootstrapping algorithm reported in [8] and described above would reveal the following properties:

1. Bootstrapping reduces the number of outliers [8]: Perhaps outliers are the main cause for the overlap between regions of different classes in the feature space. By reducing the number of outliers, the distance between two close patterns belonging to different classes might be increased to some extent. This may improve the generalization ability of the classifier.

2. Bootstrapping makes the training patterns, of a given class, come closer to the centroid or sample mean of the class. This closeness increases with the value of $k$. This is exemplified in Fig. 1. For example, consider a situation where there are only two classes W1 and W2. In each class let there be four training samples, so that the total number of training patterns is 8. These training patterns are shown in Fig. 1 by points ‘a’–‘h’. By applying the bootstrapping algorithm presented in Section 2, with the value of $k$ as 2, the bootstrap patterns are X, X, Y and Y in class W1 and U, U, V and V in class W2 as shown in the figure. In this example, four bootstrap samples are repeated. Two in class W1 viz., X and Y and two in W2 viz., U and V. This shows that after bootstrapping the original training data, some redundancy might be introduced in the bootstrapped data. Also, the centroids of original patterns a–d in W1 and their bootstrap samples

2. Notation

We use the following notation throughout the paper. The set $\mathcal{X}$ of all training patterns is represented by

$$\mathcal{X} = \{(\mathbf{X}_1, \theta_1), (\mathbf{X}_2, \theta_2), \ldots, (\mathbf{X}_N, \theta_N)\},$$

where $N$ is the cardinality of $\mathcal{X}$. Each training pattern $\mathbf{X}_i$ is a vector of dimension $D$.

$$\mathbf{X}_i = (x_{i1}, x_{i2}, \ldots, x_{id})$$

where $x_{i1}, x_{i2}, \ldots, x_{id}$ are feature values that represent training pattern $\mathbf{X}_i$. Every training pattern $\mathbf{X}_i$ in $\mathcal{X}$ is associated with a class label $\theta_i$, where $\theta_i \in \mathcal{W}$, $\forall i = 1, 2, \ldots, N$. Here, $\mathcal{W} = \{\omega_1, \omega_2, \ldots, \omega_m\}$ is the possible collection of class labels and $m$ is the number of classes under consideration. Sets of bootstrap patterns are represented by using ‘b’ as superscript. For example, $\mathcal{X}^b$ is the set of bootstrap samples of all the patterns in $\mathcal{X}$.

Note that, it is possible to partition $\mathcal{X}$ into blocks of patterns of the same class as follows:

$$\mathcal{X} = \mathcal{X}_1 \cup \mathcal{X}_2 \cup \cdots \cup \mathcal{X}_m$$

where $\mathcal{X}_i \cap \mathcal{X}_j = \phi$,

$$\forall i, j = 1, 2, \ldots, m; i \neq j,$$

$$\mathcal{X}_a = \{(\mathbf{X}_i, \omega_a) \mid \forall i = 1, 2, \ldots, |\mathcal{X}_a|\} \quad \forall a = 1, 2, \ldots, m.$$
X and Y are the same and X and Y are closer to the centroid of W1 compared to the patterns a–d. A similar behavior is exhibited by patterns of W2.

3. If the value of $k$ is equal to the number of training patterns of a class, then all the bootstrap samples obtained by applying the above algorithm is the same pattern (the centroid of the class). For example, there are four patterns in each of the two classes in the example shown in Fig. 1. If the value of $k$ is 4, then the bootstrap samples of all the four patterns in W1 are identical. From this, it can be observed that as the value of $k$ increases, the redundancy in the bootstrap data increases.

4. Data reduction

Several methods were developed for reducing the size of the training data set to decrease the classification time required by a neighborhood classifier. These include:

1. Condensed nearest neighbor [4] and
2. Exemplar selection based on a genetic algorithm (ESGA) [3].

Here, we focus our attention on reducing the size of the training data set by removing the redundancy present, if any, in the bootstrapped training data set. For this purpose, we propose two schemes based on thresholding and use them. These two schemes are:

1. Redundancy elimination by selection of threshold (REST).

We show with an example that REST is an order-dependent method. Also, we show that the MREST is an order-independent method. We compare the performance of the classifier using the reduced training set obtained by our approach with the performance of the CNN classifier and a classifier based on ESGA.

4.1. Typical and atypical patterns

In this section, we differentiate between a ‘typical pattern’ and an ‘atypical pattern’. The distinction is based on the first $k$ neighbors of a training pattern. Intuitively, a pattern is a typical pattern if all its $k$ neighbors, drawn from the training data set, are members of the same class as itself. A pattern is atypical if at least one of its $k$ neighbors is from a different class.

Let $X_i$, $i = 1, 2, ..., N$ be a pattern from $\mathcal{X}$ with class label $\omega_j$. Let $X_{i_1}, X_{i_2}, ..., X_{i_k}$ be the first $k$ neighbors of $X_i$.

**Typical pattern:** A pattern $X_i$ is said to be a typical pattern if

$$C(X_{i_\ell}) = \omega_j, \quad \forall \ell = 1, 2, ..., k,$$

where $C(X)$ denotes the class label of pattern $X$.

**Atypical pattern:** A pattern $X_i$ is said to be an atypical pattern if it is not a typical pattern. In other words,

$$C(X_{i_\ell}) \neq \omega_j \quad \text{for at least one value of } \ell, \ell = 1, 2, ..., k.$$

Next, we present an algorithm to find the set of typical patterns which is denoted by $\mathcal{T}$.

**Input:** $k, \mathcal{X}$

**Output:** $\mathcal{T}$

1. Set $\mathcal{T} \leftarrow \phi$.
2. For each $X_i \in \mathcal{X}$ do:
   1. Initialize $\text{count} \leftarrow 0$.
   2. Find the first $k$ nearest neighbors $X_{i_1}, X_{i_2}, ..., X_{i_k}$.
   3. For each $j = 1, 2, ..., k$ do:
      1. If $C(X_{i_j}) = C(X_i)$ then:
         1. $\text{count} \leftarrow \text{count} + 1$;
      2. Else:
         1. Break;
   3. If $\text{count} = k$ then:
      1. $\mathcal{T} \leftarrow \mathcal{T} \cup \{X_i\}$;
      2. Initialize $\text{count} \leftarrow 0$;

4.2. Redundancy elimination by selection of threshold

In this approach, the dissimilarity measure is the Euclidean distance between each pair of typical training patterns, that is patterns from $\mathcal{T}$. A threshold distance
\( \tau \) is selected. If a pattern \( X \) and any of its \( k \) nearest neighbors, \( Y \) are such that \( d(X, Y) < \tau \), then \( Y \) is deleted from \( \mathcal{F} \). This means that two patterns are similar within a specified threshold \( \tau \) and hence both the training patterns need not be present in the reduced training set, and hence only one is retained. The algorithm for removing similar patterns, by the threshold approach, is as follows:

**Input:** \( \tau, k, \mathcal{F} \)

**Output:** Reduced Set \( \mathcal{R}_r \)

\[
\mathcal{R}_r \leftarrow \mathcal{F};
\]

for (each \( X_i \in \mathcal{R}_r \) ) {

find the \( k \) nearest-neighbors \( X_{i1}, X_{i2}, \ldots, X_{ik} \) of \( X_i \)

for (\( j = 1, 2, \ldots, k \)) {

if (\( d(X_i, X_{ij}) < \tau \)) {

\( \mathcal{R}_r \leftarrow (\mathcal{R}_r \setminus \{X_{ij}\}) \);

}

}

}

Selecting an appropriate \( \tau \) plays a crucial role in reducing the size of the training data set. If the value of \( \tau \) is too high, the training set size will become very small and the performance of the classifier might suffer. If the value of \( \mathcal{R}_r \) is too low, the reduction in the training data set size will be not be significant. We show, with a simple example, that this algorithm is an order-dependent algorithm. Fig. 2 shows three points A–C with coordinates (1,2), (4,2) and (7,2), respectively. If the order presented to the above algorithm is A–C, respectively, with \( k = 2 \) and \( \tau = 3.5 \), then this algorithm will select \( \{A, C\} \) as prototype set. If the order is changed form A, B, C to B, A, C, with the same parameter values, the output of this algorithms is \( \{B\} \). This clearly shows that the above algorithm is an order-dependent algorithm.

### 4.3. Modified redundancy elimination by selection of threshold

In this subsection we present a modified version of REST that selects exemplars from the bootstrapped data set. The difference between the REST presented in Section 4.2 and this algorithm is that REST is an order-dependent algorithm, whereas MREST is an order-independent algorithm. In this subsection we present the algorithm and also show that MREST is an order-independent algorithm.

**Algorithm.** Let at level \( \ell = 1 \) \( P'_p = \{X_r : \forall r = 1, 2, \ldots, m \} \)

Where \( P'_p \) is the \( p \)th permutation of the training set \( \mathcal{X} \), at level \( \ell \); Reduced Set \( \mathcal{R} = \phi \)

1. Calculate the mean \( \mu'_p \) of the patterns in set \( P'_p \).

   \[
   \mu'_p = \sum_{i=1}^{|P'_p|} X_i / |P'_p|
   \]

2. Find \( \delta(P'_p, \mu'_p) \), the nearest neighbor to the mean \( \mu'_p \).

   \if more than one pattern is having minimum distance to \( \mu'_p \) (say \( s \) patterns, \( X_{r_1}, \ldots, X_{r_s} \)) \{

   \begin{align*}
   \text{sort all such patterns based on a non-decreasing order of feature values. */}
   
   X_{min} = X_{r_1},
   
   \text{for(each } k = 2, 3, \ldots, s \text{) } \{
   
   \text{for(each } d = 1, 2, \ldots, D \text{) } \{
   
   \text{if } ( X_{rs} < X_{min} ) \{
   
   X_{min} = X_{rs},
   
   \text{break;}
   
   \}
   
   \}
   
   \}
   
   \}
   
   \}
   
   \}
   
   \}
   
   \}
   
   \}
   
   \}
   
   \}
   
   \}
   
   \}
   
   \}
   
   \}
   
   \}
   
   \}
   
   \}
   
   \}
   
   \}

3. Reduced Set \( \mathcal{R} = \mathcal{R} \cup \{ \delta(P'_p, \mu'_p) \} \)

4. for (\( j = 1, 2, \ldots, |P'_p| \)) {

   \text{if } (d(\delta(P'_p, \mu'_p), X_j) <= \tau) \{

   \begin{align*}
   P'_p = P'_p \setminus \{X_j\}
   
   \}
   
   \}
   
   \}
   
   \}

5. \( P'_p = P'_p \setminus \{ \delta(P'_p, \mu'_p) \} ; \ell = \ell + 1; \)

6. if (|\( P'_p \)| \( \neq 0 \))

   \begin{align*}
   \text{goto step 1;}
   
   \end{align*}

   \text{else}

   \begin{align*}
   \text{Terminate;}
   
   \end{align*}

It can be observed that the above algorithm terminates in at most \( |P'_p| \) steps for a threshold value \( \tau > 0 \).

**Theorem 1.** The algorithm MREST is order independent.

**Proof.** Let two permutations \( P'_p, P'_q \) (where \( p \neq q \)) of the training data set \( \mathcal{X}, \forall r = 1, 2, \ldots, m \) (\( \mathcal{X} \subset \mathcal{X} \)) be presented to the above algorithm. \( P'_p \neq P'_q \) by viewing \( P'_p, P'_q \) as ordered sets. However, if they are viewed as sets, then \( P'_p = P'_q \). That is, the order of the elements in the permuted sets \( P'_p, P'_q \) may be different but the elements in the two permuted sets are the same. Also, \( P'_p = \mathcal{X} \forall p = 1, 2, \ldots, |P'_p| \) at \( \ell = 1 \).
To show that the algorithm selects the same exemplars for any two permuted sets $P_p', P_q'$ of the training data set $\mathcal{X}$, it would be enough if we show that

1. At every level $\ell$, $\mu_p' = \mu_q'$ for all $p, q$ and
2. That $\delta'(P_p', \mu_p') = \delta'(P_q', \mu_q')$ at every level.

This is because, the reduced set $\mathcal{R}$ is updated only in step 3 of the algorithm and this updation depends on $\delta(P_p, \mu_p')$, which in turn depends on $\mu_p'$.

At level $\ell = 1$, the means of the permuted sets $P_p', P_q'$ are given by

$$
\mu_p' = \frac{X_i + \sum_{j \neq i} X_j}{|P_p'|}, \\
\mu_q' = \frac{X_j + \sum_{i \neq j} X_i}{|P_q'|}.
$$

It can be observed from the above two equations that $\mu_p'$ and $\mu_q'$ are equal (since the numerators and denominators of the right-hand sides of the two equations are equal). It is possible to show Eq. (2) above based on the proof of the lemma given below. □

**Lemma 1.** The nearest neighbor of the mean from two permuted sets is the same.

**Proof.** The nearest neighbor of the mean is that pattern which has same minimum distance to the mean compared to all the other patterns in the training data set. If more than one pattern has minimum distance to the mean, then there occurs a tie. The tie is resolved in step 2 of MREST algorithm and this step ensures uniqueness.

Therefore, $\delta'(P_p', \mu_p') = \delta'(P_q', \mu_q')$. Threshold method always deletes the same patterns, since $d(\delta'(P_p', \mu_p'), X_i) \forall X_i \neq \delta'(P_p', \mu_p')$ is same for all permutations. Therefore, step 4 in MREST algorithm always yields the same result that is, $P_p', P_q'$ are same.

Hence, at the end of level $\ell = 1$, $P_p' = P_q'$. The above argument is true at every level, since the mean is the same and its nearest neighbor selected is the same. Therefore, the updation of the reduced set $\mathcal{R}$ yields the same result. □

Two extreme cases can be considered in the threshold method.

1. Threshold value is too small. If
   $$d[\delta'(P_p', \mu_p'), X_i] > \tau \forall r = 1, 2, \ldots, N,$$
   and $r \neq \delta'(P_p', \mu_p')$,

   then, no pattern is deleted from the training data set.

2. Threshold value is too large. If
   $$d[\delta'(P_p', \mu_p'), X_i] < \tau \forall r = 1, 2, \ldots, N,$$
   and $r \neq \delta'(P_p', \mu_p')$,

   then, all the patterns will be deleted from the training data set $\delta'(P_p', \mu_p')$ is included in the reduced set $\mathcal{R}$.

5. **Data set**

In the present work we use a handwritten digit data set in our experiments. The handwritten numeral data set contains patterns belonging to 10 different classes, one class corresponding to each of the digits ‘0’–‘9’. Each pattern is a binary image of size $32 \times 24$ pixels. Therefore, the dimensionality of each pattern is 768. To reduce the dimensionality, non-overlapping windows of size $2 \times 2$ are formed over the entire image and each window is replaced by one feature whose value corresponds to the number of ‘on’ pixels in the considered window. So, the value of each pixel ranges from 0 to 4. The dimensionality of each pattern reduces by a factor 4 and is equal to 192 ($16 \times 12$). The training data set contains 6670 patterns consisting of 667 samples from each class. The test data set contains 3333 patterns. The number of test patterns in each class is approximately 333. We have used the bootstrapping algorithm on this data set, with the value of $k$ equal to 16 (This value of 16 has been arrived at by increasing the value of $k$ from 5 to 25, testing on the validation data set and by taking the optimal value of $k$, that is the value of $k$ for which the classification accuracy on the validation data is the maximum).

6. **A comparative study**

We have compared REST and MREST with the condensed nearest neighbor (CNN) and ESGA. The CNN is one of the most popularly used methods for reducing the size of the training set. This method has been well described in the literature [4]. Another approach is to pose the exemplar selection problem as an optimization problem and select the prototypes using an optimization tool, for example, the Genetic Algorithm (GA) [9]. Exemplar selection using GAs (ESGA) was first used in Ref. [3]. A brief discussion of selecting prototypes using GAs is presented in the next subsection.

6.1. **Exemplar selection using GAs**

In this section, we describe ESGA for selecting prototypes. GAs are robust and are not likely to be affected in the presence of local optima in the solution space. They concentrate on a set of promising solutions that may
converge to global optimum solution. GAs start with a population of solution strings called chromosomes. Each chromosome is assigned a value known as fitness value, using a fitness function $F$. The fitness function can be considered to be the same as the objective function. The chromosomes which are fitter will participate actively to generate next population of chromosomes. In every generation, three genetic operators viz., selection, crossover and mutation are used on the population of GA. The essential idea in using the selection operator or reproduction operator is that chromosomes having better fitness value are picked from the current population with a higher probability. The probability of selecting the $i$th chromosome is calculated as

$$p_i = \frac{F_i}{\sum_{j=1}^{n} F_j}.$$ 

In reproduction operation no new chromosomes are generated. Crossover operator is mainly responsible for the generation of new chromosomes. The mutation operator changes, in a chromosome, the value from 1 to 0 and vice versa with a small mutation probability $p_m$ in each position of a chromosome. A good overview of GAs is provided by Goldberg [9]. Chang and Lippmann [3] used GAs for selecting an optimal subset of patterns (prototypes) from the training set.

### 6.1.1. Encoding and fitness function

In GAs, a chromosome is a binary string that represents a subset of patterns. The length of a chromosome, for prototype selection, depends on the number of training patterns. Presence of ‘1’ in location $i$ of chromosome means that the training pattern $i$ is included in the prototype set, and presence of ‘0’ in the $i$th location of the chromosome indicates the absence of the $i$th training pattern. The classification accuracy of the NNC is used as fitness function for evaluating the chromosome. The classification accuracy is computed as

$$\text{Classification accuracy} = \frac{\text{Number of patterns correctly classified}}{\text{Total number of patterns}} \times 100.$$ 

### 6.1.2. How GAs are used in prototype selection?

After having chosen the coding scheme and the fitness function, a random population of chromosomes is generated. The fitness values for all the chromosomes in the population are calculated. The selection operator is then applied to reproduce those chromosomes which have a higher probability of selection. The crossover and mutation operators are then used on these chromosomes and the process is repeated till the termination criterion is reached. Selection of prototypes using GAs is well described by Chang and Lippmann [3].

![Fig. 3. Prototype selection model.](image)

### Table 1

<table>
<thead>
<tr>
<th>Bootstrapping with $k$ value</th>
<th>Classification accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>96.40</td>
</tr>
<tr>
<td>10</td>
<td>96.70</td>
</tr>
<tr>
<td>15</td>
<td>96.80</td>
</tr>
<tr>
<td>16</td>
<td>97.00</td>
</tr>
<tr>
<td>18</td>
<td>96.60</td>
</tr>
<tr>
<td>20</td>
<td>96.90</td>
</tr>
<tr>
<td>25</td>
<td>96.70</td>
</tr>
</tbody>
</table>

A general prototype selection model is presented in Fig. 3. The input to this algorithm is the training data set. The output given by this algorithm is the prototype set.

### 6.2. Experimental results

The bootstrapping algorithm is applied with different values of $k$ and the corresponding results are presented in Table 1 using the validation data set. From this table we note that, the classification accuracy attains a peak value for some intermediate value of $k$ in the range considered. This is mainly due to the phenomenon that by averaging $k$ neighboring patterns, where $k$ is large enough, the resultant pattern might become a corrupted version of the original character. We have chosen the value of $k$ as 16 for conducting all the experiments because, for this value of $k (= 16)$ yields maximum classification accuracy on the training data and validation data sets.

Experiments are conducted on both the original and bootstrapped data sets. We use the following algorithms to carry out the experiments.

- ESGA-based classifier.
- Condensed nearest-neighbor classifier.
- REST-based classifier.
- MREST-based classifier.

### 6.2.1. ESGA-based classifier

For conducting experiments, we have chosen the following parameters for the GAs. The initial population of size 10 is generated randomly. We chose a small size population because of the practical difficulty in evaluating the fitness function. The length of the chromosome is...
the same as the total number of training patterns. That is, the length of chromosome = |D|. We terminate the run of GA when the maximum number of generations exceeds 5000. We have chosen two-point crossover and binary tournament selection as the genetic operators. The crossover probability is 0.95 and mutation probability is 0.009. We have chosen these values based on initial experiments. For training the GA, we have used the validation data set consisting of 1000 patterns. At the end of 5000 generations, we identify the best chromosome and the test data is tested on this chromosome. At the end of 5000 generations, 3364 prototypes were selected by the Genetic algorithm using original data. The classification accuracy found using the selected prototypes is 94.03%. In the case of bootstrapped samples, 3308 prototypes were selected with a classification accuracy of 96.46%.

6.2.2. CNN classifier
Condensed nearest neighbor is used both on the original data set and on the bootstrapped data set. In the case of original data set, 1174 prototypes were selected. Using these as training patterns, the classification accuracy using test data set was found to be 92.45%. On the other hand, 121 prototypes were selected using the bootstrapped data set. The classification accuracy using these 121 prototypes is found to be 91.24%. This shows that bootstrapping has the potential to reduce classification time without degrading classification accuracy significantly.

6.2.3. Classifier based on REST and MREST algorithms
We have conducted experiments with different values of threshold for the two algorithms and the results are presented in Tables 2 and 3. As the threshold value \( q \) increases, the number of prototypes selected were observed to be decreasing (see Figs. 4 and 5). Fig. 4 shows the dependency of the size of the bootstrapped data set and classification accuracy on the threshold value \( q \). Fig. 5 is obtained by conducting experiments on the original data set. It can be observed from the plot that the classification accuracy is not varying significantly even when the training set size is reduced considerably.

An interesting point to be observed here is that even if the number of training patterns is reduced nearly by

Table 2
Classification accuracy using REST

<table>
<thead>
<tr>
<th>Threshold value</th>
<th>Bootstrapped data set</th>
<th>Original data set</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Number of patterns</td>
<td>Classification accuracy (%)</td>
</tr>
<tr>
<td>1.0</td>
<td>6631</td>
<td>96.49</td>
</tr>
<tr>
<td>2.0</td>
<td>6519</td>
<td>96.49</td>
</tr>
<tr>
<td>3.0</td>
<td>6163</td>
<td>96.46</td>
</tr>
<tr>
<td>4.0</td>
<td>5158</td>
<td>96.40</td>
</tr>
<tr>
<td>5.0</td>
<td>3740</td>
<td>96.34</td>
</tr>
<tr>
<td>6.0</td>
<td>2529</td>
<td>96.25</td>
</tr>
<tr>
<td>7.0</td>
<td>1942</td>
<td>96.31</td>
</tr>
<tr>
<td>8.0</td>
<td>1756</td>
<td>96.07</td>
</tr>
<tr>
<td>9.0</td>
<td>1712</td>
<td>96.07</td>
</tr>
<tr>
<td>10.0</td>
<td>1693</td>
<td>96.04</td>
</tr>
</tbody>
</table>

Table 3
Classification accuracy using MREST

<table>
<thead>
<tr>
<th>Threshold value</th>
<th>Bootstrapped data set</th>
<th>Original data set</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Number of patterns</td>
<td>Classification accuracy (%)</td>
</tr>
<tr>
<td>1.0</td>
<td>6078</td>
<td>96.52</td>
</tr>
<tr>
<td>2.0</td>
<td>1593</td>
<td>95.98</td>
</tr>
<tr>
<td>3.0</td>
<td>458</td>
<td>92.50</td>
</tr>
<tr>
<td>4.0</td>
<td>56</td>
<td>84.73</td>
</tr>
<tr>
<td>5.0</td>
<td>20</td>
<td>74.98</td>
</tr>
</tbody>
</table>
a factor of 2, the classification accuracy obtained using the REST is still very high. For example, from Table 2 it can be observed that, for the threshold value 6.0, the classification accuracy with 2529 prototypes is 96.25%. And when the threshold value is 7.0, the classification accuracy with 1942 prototypes is 96.31%. This means that the classification accuracy does not decrease with the value of threshold. Moreover, for approximately the same number of training patterns retained, MREST performs better than REST. For example, (see Tables 2 and 3) for 6519 prototypes selected using REST, the classification accuracy is 96.49%, whereas for 6078 prototypes, selected using MREST, the classification accuracy is 96.52%. Also, we can observe that, use of MREST results in maximum classification accuracy. The classification accuracy is decreasing with an increase in the threshold value in the case of MREST. But for an appropriate threshold level, the number of prototypes selected and the classification accuracy obtained are encouraging as can be observed from Table 3.

6.3. Effect of typical and atypical patterns

Here we make a few comments on the number of typical and atypical patterns obtained from the bootstrapped data as well as from the original data set.

1. As the value of $k$, the number of nearest neighbors defining the typical patterns, increases the number of typical patterns decreases (refer to Table 4).
Table 5
Effect of atypical patterns on classification accuracy

<table>
<thead>
<tr>
<th>k value</th>
<th>No. of atypical patterns deleted from original data set</th>
<th>Classification accuracy (%)</th>
<th>No. of atypical patterns deleted from bootstrapped patterns</th>
<th>Classification accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>841</td>
<td>93.97</td>
<td>4</td>
<td>96.49</td>
</tr>
<tr>
<td>10</td>
<td>1783</td>
<td>92.14</td>
<td>17</td>
<td>96.55</td>
</tr>
<tr>
<td>15</td>
<td>2452</td>
<td>90.88</td>
<td>39</td>
<td>96.52</td>
</tr>
<tr>
<td>20</td>
<td>2972</td>
<td>88.90</td>
<td>70</td>
<td>96.46</td>
</tr>
<tr>
<td>25</td>
<td>3381</td>
<td>86.59</td>
<td>132</td>
<td>96.34</td>
</tr>
<tr>
<td>30</td>
<td>3701</td>
<td>85.63</td>
<td>185</td>
<td>96.25</td>
</tr>
</tbody>
</table>

Bootstrapped patterns were obtained for a fixed value of $k = 16$. Typical and atypical patterns are then obtained varying the value of $k$ from 5 to 30 on these bootstrapped patterns.

2. The number of typical patterns, after bootstrapping, is not less than that of the typical patterns in the original data set (refer Table 4). If let the number of typical patterns obtained after bootstrapping is $M_b$ and the number of typical patterns obtained in the original data set be $M$, then $M_b \geq M$.

Table 4 is based on the results obtained by conducting experiments for finding typical and atypical patterns present in the original data set and after bootstrapping. It experimentally shows that the number of typical patterns after bootstrapping is larger than the number of typical patterns in the original data.

From the results presented in Table 5, it can be observed that, atypical patterns may not help in improving the classification accuracy. In other words, the classifier’s performance (in terms of classification accuracy) may not degrade significantly even in the absence of the atypical patterns, and hence these may be deleted from the bootstrapped data set.

6.3.1. Second level bootstrapping

In this section we study the effect of one more level (second) of bootstrapping. Second level bootstrapped samples are obtained by applying bootstrap algorithm on the first level bootstrapped patterns.

Fig. 6 shows the effect of a second-level bootstrapping. The bootstrapped samples are obtained using different values of $k$ on the original training data set. The second-level bootstrap samples are obtained by using the same value of $k$ on the bootstrapped samples. From this plot, we can observe that second-level bootstrap does not yield a comparable classification accuracy.

<table>
<thead>
<tr>
<th>Data Set</th>
<th>Typical patterns</th>
<th>Atypical patterns</th>
<th>k value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original</td>
<td>5501</td>
<td>1169</td>
<td>5</td>
</tr>
<tr>
<td>bootstrapped</td>
<td>6666</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>Original</td>
<td>4559</td>
<td>2111</td>
<td>10</td>
</tr>
<tr>
<td>bootstrapped</td>
<td>6653</td>
<td>17</td>
<td>10</td>
</tr>
<tr>
<td>Original</td>
<td>3890</td>
<td>2780</td>
<td>15</td>
</tr>
<tr>
<td>bootstrapped</td>
<td>6631</td>
<td>39</td>
<td>15</td>
</tr>
<tr>
<td>Original</td>
<td>3370</td>
<td>3300</td>
<td>20</td>
</tr>
<tr>
<td>bootstrapped</td>
<td>6600</td>
<td>70</td>
<td>20</td>
</tr>
<tr>
<td>Original</td>
<td>2961</td>
<td>3709</td>
<td>25</td>
</tr>
<tr>
<td>bootstrapped</td>
<td>6538</td>
<td>132</td>
<td>25</td>
</tr>
<tr>
<td>Original</td>
<td>2641</td>
<td>4029</td>
<td>30</td>
</tr>
<tr>
<td>bootstrapped</td>
<td>6485</td>
<td>185</td>
<td>30</td>
</tr>
</tbody>
</table>
7. Conclusions

In this paper, we have presented two threshold-based algorithms for selecting prototypes viz., REST and MREST from bootstrapped data set. REST is order dependent, whereas MREST is an order-independent algorithm. We had shown the order-independence of the MREST algorithm. The performance of both the algorithms were compared with that of the traditional methods, viz., CNN and prototype selection using genetic algorithms. It can be observed from the results that the threshold-based redundancy removal schemes using bootstrapped data set perform better than existing methods. The performance the classifier based on MREST is better than that of the classifier based on REST for approximately the same number of prototypes selected.

Even though bootstrapping has high potential in improving the classification accuracy, the redundancy present in the data suggests that the entire data need not be used for classification. An important result of the experiments is that the number of typical patterns after bootstrapping is more than the number of typical patterns in the original data set. Also, higher levels of bootstrapping do not improve the classification accuracy.

Acknowledgements

The authors thank Professor U.R. Prasad, Department of Computer Science and Automation, Indian Institute of Science, Bangalore, India for his critical comments on an earlier version of this paper. The comments of the anonymous referee have helped in enhancing the quality of the paper.

References


About the Author—V. VIJAYA SARADHI received his B. Tech. degree in Civil Engineering from the Nagarjuna University, Guntur, India and M.S. degree from the Department of Computer Science and Automation, Indian Institute of Science, Bangalore, India. His research interests include Pattern Recognition, Genetic Algorithms, and Neural Networks.

About the Author—M. NARASIMHA MURTY received his B.E., M.E. and Ph.D. degrees from the Indian Institute of Science, Bangalore, India. He is currently a Professor in the Department of Computer Science and Automation. His research interests include Pattern Recognition, Genetic Algorithms and Data Mining.