Recursive Prototype Reduction Schemes Applicable for Large Data Sets

Sang-Woon Kim[1] and B. J. Oommen[2]

[1] Member IEEE.
Div. of Computer Science and Engineering, Myongji University
Yongin, 449-728 Korea
kimsw@mju.ac.kr
[2] Senior Member IEEE.
School of Computer Science, Carleton University
Ottawa, ON, K1S 5B6, Canada
oommen@scs.carleton.ca

Abstract. Most of the Prototype Reduction Schemes (PRS), which have been reported in the literature, process the data in its entirety to yield a subset of prototpyes that are useful in nearest-neighbour-like classification. Foremost among these are the Prototypes for Nearest Neighbour (PNN) classifiers, the Vector Quantization (VQ) technique, and the Support Vector Machines (SVM). These methods suffer from a major disadvantage, namely, that of the excessive computational burden encountered by processing *all* the data. In this paper, we suggest a recursive and computationally superior mechanism. Rather than process all the data using a PRS, we propose that the data be recursively subdivided into smaller subsets. This recursive subdivision can be arbitrary, and need not utilize any underlying clustering philosophy. The advantage of this is that the PRS processes *subsets* of data points that effectively sample the entire space to yield smaller subsets of prototypes. These prototypes are then, in turn, gathered and processed by the PRS to yield more refined prototypes. Our experimental results demonstrate that the proposed recursive mechansim yields classification comparable to the *best* reported prototype condensation schemes to-date, for both artificial data sets and for samples involving real-life data sets. The results especially demonstrate the computational advantage of using such a recursive strategy for *large* data sets, such as those involved in data mining and text categorization applications.

1 Introduction

In non-parametric pattern classification like the nearest neighbour (NN) or the k-nearest neighbour ($k-$NN) rule, each class is described using a set of sample prototypes, and the class of an unknown vector is decided based on the identity of the closest neighbour(s) which are found among all the prototypes [1]. For applications, involving large data sets, such as those involved in data mining, financial forecasting, retrieval of multimedia databases and biometrics, it is advantageous to reduce the number of training vectors while simultaneously insisting that the classifiers that are built on the reduced design set perform as well, or nearly as well, as the classifiers built on the original data set. Various prototype reduction schemes, which are useful in nearest-neighbour-like classification, have been reported in the literature [2].

One of the firsts of its kind is the Condensed Nearest Neighbour (CNN) rule [3]. The reduced set produced by the CNN, however, customarily includes "interior" samples, which can be completely eliminated, without altering the performance of the resultant classifier. Accordingly, other methods have been proposed successively, such as the Reduced Nearest Neighbour (RNN) rule [4], the Prototypes for Nearest Neighbour (PNN) classifiers [5], the Selective Nearest Neighbour (SNN) rule [6], two modifications of the CNN [7], the Edited Nearest Neighbour (ENN) rule [8], and the non-parametric data reduction method [9]. Besides these, in [10], the Vector Quantization (VQ) and the Bootstrap [11] techniques have also been reported as being extremely effective approaches to data reduction. Recently, Support Vector Machines (SVM) [12] have proven to possess the capability of extracting vectors that support the boundary between the two classes. Thus, they have been used satisfactorily to represent the global distribution structure. Recently, we have proposed a new hybrid scheme, the Kim_Oommen Hybridized Technique, [13], which is based on the philosophy of invoking *creating* and *adjusting* phases to the prototype vectors. First, a reduced set of initial prototype vectors is chosen by any of the previously mentioned methods, and then their optimal positions are learned with an LVQ3-type algorithm, thus, minimizing the average classification error. The details of this are omitted here.

All the PRS methods reported in the literature (including the one proposed in [13]) are practical as long as the size of the data set is not "too large". The applicability of these schemes for large-sized data set is limited because they all suffer from a major disadvantage – they incur an excessive computational burden encountered by processing *all the data points.* To overcome this disadvantage for large-sized data sets, in this paper, we suggest a recursive mechanism. Rather than process all the data using a PRS, we propose that the data be recursively subdivided into smaller subsets. *We emphasize that the smaller subsets need* **not** *represent sub-clusters of the original data set.* After this recursive subdivision, the smaller subsets are reduced with a PRS. The resultant sets of prototypes obtained are, in turn, gathered and processed at the higher level of the recursion to yield more refined prototypes. This sequence of divide-reduce-coalesce is invoked recursively to ultimately yield the desired reduced prototypes. We refer to the algorithm presented here as the *Kim_Oommen_Recursive PRS.*

The main contribution of this paper is the demonstration that the speed of data condensation schemes can be increased by recursive computations - which is crucial in large-sized data sets. This has been done by introducing the *Kim_Oommen_Recursive PRS*, and demonstrating its power in both speed and accuracy. We are not aware of any similar reported recursively-motivated PRS.

The first outstanding advantage of this mechanism is that the PRS can select more refined prototypes in significantly less time. This is achieved without sacrificing the classification accuracy and the prototype reduction rate. This is primarily because the new scheme does **not** process all the data points at *any* level of the recursion. A second advantage of this scheme is that the recursive subdivision can be arbitrary, and *need not utilize any clustering philosophy.* Furthermore, *all* subsequent recursive partitionings, also, need not involve clustering. Finally, the higher level PRS invocations do not involve any points interior to the Voronoi space because they are eliminated at the leaf levels.

The reader should observe that this philosophy is *quite distinct* from the partitioning using clustering methods which have recently been proposed in the literature to solve the Travelling Salesman Problem (TSP) [15]. The differences between these two philosophies can be found in [16].

The experimental results on synthetic and real-life data prove the power of these enhancements. The real-life experiments include three "medium-size" data sets, and two *large* data sets with a fairly high dimensionality. The results are conclusive.

2 Prototype Reduction Schemes

As mentioned previously, various Prototype Reduction Schemes (PRS) have been proposed in the literature - a survey of which is found in [2]. The most pertinent ones are reviewed here by two groups: the conventional methods and a newly proposed hybrid method. Among the conventional methods, the CNN and the SVM are chosen as representative schemes of *selecting* methods. The former is one of first methods proposed, and the latter is more recent. As opposed to these, the PNN and VQ (or SOM) are considered to fall within the family of prototype-*creating* algorithms. The reviews of these methods is not attempted here. However, it should be emphasized that our new recursive method can utilize *any one* of the reported PRS as an atomic building block – thus extending them. In that light, many of the PRS are *briefly* surveyed in the unabridged version of the paper [16]. This present study (and so the review in [16]) includes the CNN rule [3], the PNNs [5], the VQ and SOM methods [14], and the SVM [12].

The unabridged paper [16] also describes a newly-reported hybrid scheme, the Kim_Oommen Hybridized Technique, [13], which is based on the philosophy of invoking *creating* and *adjusting* phases. First, a reduced set of initial prototypes or code-book vectors is chosen by any of the previously mentioned methods, and then their optimal positions are learned with an LVQ3-type algorithm, thus, minimizing the average classification error.

3 Recursive Invocations of PRS

3.1 The Rationale of the Recursive Algorithm

Since prototypes near the boundary play more important roles than the interior ones for designing NN classifiers, the points near the boundary are more important in selecting the prototypes. In all the currently reported PRS, however, points in the interior of the Voronoi space are processed for, apparently, no reason. Consequently, all reported PRS suffer from an excessive computational burden encountered by processing all the data, which becomes very prominent in "large" data sets.

To overcome this disadvantage, in this section, we propose a recursive mechanism, where the data set is sub-divided recursively into smaller subsets to filter out the "useless" internal points. Subsequently, a conventional PRS processes the smaller subsets of data points that effectively sample the entire space to yield *subsets* of prototypes – one set of prototypes for each subset. The prototypes, which result from each subset, are then coalesced, and processed again by the PRS to yield more refined prototypes. In this manner, prototypes which are in the interior of the Voronoi boundaries, and are thus ineffective in the classification, are eliminated at the subsequent invocations of the PRS.

A direct consequence of eliminating the "redundant" samples in the PRS computations, is that the processing time of the PRS is *significantly* reduced. This will be clarified by the example below.

To illustrate the functioning of the recursive process, we present an example for the two-dimensional data set referred to as "Random". Two data sets, namely the training and test sets, are generated randomly with a uniform distribution, but with irregular decision boundaries. The training set of 200 sample vectors is used for computing the prototypes, and the test set of 200 sample vectors is used for evaluating the quality of the extracted prototypes.

To demonstrate the power of the mechanism, we first select prototypes from the whole training set using the CNN method. This is also repeated after randomly dividing the training set into two subsets of equal size – each with 100 vectors. Fig. 1 shows the whole set of the "Random" training data set, the divided subsets and the prototypes selected with the CNN and the recursive PRS methods, respectively. Observe that in this example, to render the presentation brief, we have invoked the recursive procedure *only* twice.

In Fig. 1, the set of prototypes of (e), which is extracted from the whole set of (a), consists of 36 points and has a classification accuracy of 96.25 % . The prototypes of (f) and (g), selected from the subsets of (b) and (c), *both* consist of the 21 vectors, and have accuracies of 96.00 % and 94.50 % respectively.

On the other hand, the set of prototypes of (h), which is extracted from a data set obtained by combining two prototype sets of (f) and (g), consists of 27 points, and has the accuracy of 97.00 %. Moreover, it should be pointed out that the time involved in the prototype selection of (h), is *significantly* less than that of (e), because the number of sample vectors of the combined sets of (f) and (g) together, is smaller than that of the whole set of (a).

The formal algorithm is omitted here in the interest of brevity, but can be found in [16]. However, a *brief* explanation of the algorithm is not out of place. If the size of the original data set is smaller than K, a traditional PRS is invoked to get the reduced prototypes. Otherwise, the original data set is recursively subdivided into J subsets, and the process continues down towards

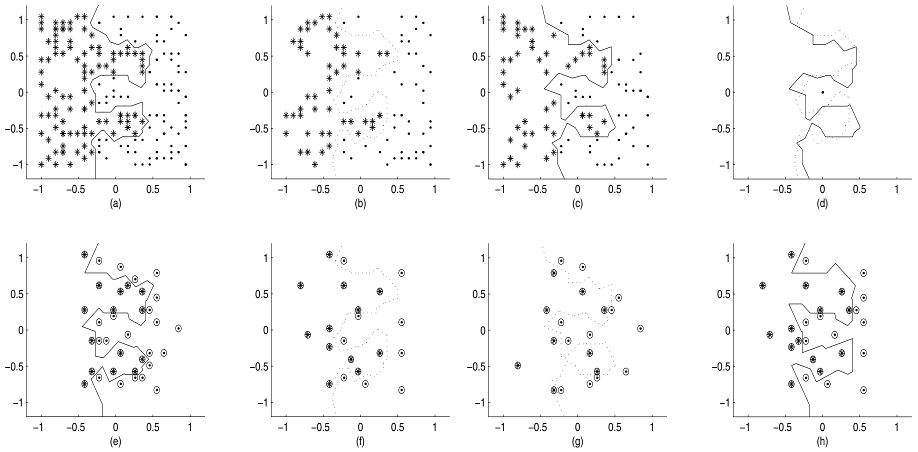

Fig. 1. The entire set, the divided subsets and the prototypes of the "Random" data set, where the vectors of each class are represented by '$*$' and '$\cdot$', and the selected prototype vectors are indicated by the circled '$*$' and '$\cdot$' respectively. The details of the figure are found in the text

the leaf of the recursive tree. Observe that a traditional PRS is invoked only when the corresponding input set is "small enough". Finally, at the tail end of the recursion, the resultant output sets are merged, and if *this* merged set is greater than K the procedure is again recursively invoked.

It should be also noted that the traditional PRS, which can be otherwise time consuming for large data sets, is never invoked for *any* sets of cardinality larger than K. It is invoked *only* at the leaf levels when the sizes of the sets are "small", rendering the entire computation very efficient.

4 Experimental Results

4.1 Experimental Results: Medium-Sized Data Sets

The Kim_Oommen_Recursive PRS has been rigorously tested and compared with many conventional PRS. This was first done by performing experiments on a number of "medium-sized" data sets, both real and artificial. The data set named "Non_normal (Medium-size)", which has been also employed in [9], [10] and [11] as a benchmark experimental data set, was generated from a mixture of four 8-dimensional Gaussian distributions as follows:

1. $p_1(x) = \frac{1}{2}N(\mu_{11}, I_8) + \frac{1}{2}N(\mu_{12}, I_8)$ and
2. $p_2(x) = \frac{1}{2}N(\mu_{21}, I_8) + \frac{1}{2}N(\mu_{22}, I_8)$,

where $\mu_{11} = [0, 0, \cdots, 0]$, $\mu_{12} = [6.58, 0, \cdots, 0]$, $\mu_{21} = [3.29, 0, \cdots, 0]$ and $\mu_{22} = [9.87, 0, \cdots, 0]$. Here, I_8 is the 8-dimensional *Identity* matrix.

The "Sonar" data set contains 208 vectors. Each sample vector, of two classes, has 60 attributes which are all continuous numerical values. The "Arrhythmia" data set contains 452 patterns with 279 attributes, 206 of which are real-valued, and the rest are nominal. The details of these sets are found in [16] and omitted here in the interest of brevity. However, we mention that both the data sets "Sonar" and "Arrhythmia" are real benchmark data sets, cited from the UCI Machine Learning Repository [17].

In the above data sets, all of the vectors were normalized within the range $[-1, 1]$ using their standard deviations. Also, for every class j, the data set for the class was randomly split into two subsets, $T_{j,t}$ and $T_{j,V}$, of equal size. One of them was used for choosing initial code-book vectors and training the classifiers as explained above, and the other subset was used in the validation (or testing) of the classifiers. The roles of these sets were later interchanged.

In this case, because the size of the sets was not excessively large, the recursive versions of CNN, PNN, VQ and SVM, were all invoked only for a depth of *two*. The experimental results of the CNN, PNN, VQ and SVM methods implemented with the recursive mechanism, for the "Non_normal (Medium-size)" data sets is shown in Table 1. The results for the data sets "Sonar" and "Arrhythmia" are not included here in the interest of brevity. They can be found in [16].

Table 1. The experimental results of the recursive CNN, PNN, VQ and SVM methods for the "Non_normal (Medium-size)" data set. Here, *DS*, *CT*, *NP* and *Acc* are the data set size (the number of sample vectors), the processing CPU-time (in seconds), the number of prototypes, and the classification accuracy rate (%), respectively

Methods	*DS1*	*CT1*	*NP1*	*Acc1*	*DS2*	*CT2*	*NP2*	*Acc2*
CNN	500	0.61	64	92.60	500	0.63	66	91.20
	250	0.13	34		250	0.15	37	
	250	0.15	34		250	0.12	29	
	68	0.11	54	91.60	66	0.11	54	90.00
PNN	500	81.74	56	92.40	500	208.55	380	91.60
	250	7.58	31		250	7.55	34	
	250	7.54	29		250	7.54	26	
	60	0.22	46	89.20	60	0.26	50	88.80
VQ	500	0.23	4	95.60	500	0.41	4	94.80
	250	0.11	4		250	0.15	4	
	250	0.16	4		250	0.13	4	
	8	0.06	4	95.60	8	0.07	4	94.40
SVM	500	0.86	62	95.40	500	0.97	57	94.40
	250	0.24	32		250	0.25	35	
	250	0.23	30		250	0.24	24	
	62	0.07	60	95.60	59	0.06	57	94.40

The Kim_Oommen_Recursive PRS can be compared with the non-recursive versions using three criteria, namely, the processing CPU-time (*CT*), the classification accuracy rate (*Acc*), and the prototype reduction rate (*Re*).

We report below a summary of the results obtained for the case when one subset was used for training and the second for testing. The results when the roles of the sets are interchanged are almost identical. From Table 1, we can see that the *CT* index (the processing CPU-time) of the pure CNN, PNN, VQ and SVM methods can be reduced significantly by merely employing the recursive philosophy.

Consider the PNN method for the "Non_normal (Medium-size)" data set. If the 500 samples were processed non-recursively, the time taken is 81.74 seconds, the size of the reduced set is 56, and the resulting classification accuracy is 92.4%. However, if the 500 samples are subdivided into two sets of 250 samples each, processing each subset involves only 7.58 and 7.54 seconds leading to 31 and 29 reduced prototypes respectively. When *these* 60 samples are, in turn, subjected to a pure PNN method, the number of prototype samples reduced to 46 in just 0.22 seconds and yielded an accuracy of 89.2 %. If we reckon that the recursive computations can be done in parallel, the time required is only *about one-tenth* of the time which the original PNN would take. Even if the computations were done serially, the advantage is marked. Such results are typical, as can be seen from [16].

4.2 Experimental Results: Large-Sized Data Sets

In order to further investigate the advantage gained by utilizing the proposed recursive PRS for more computationally intensive sets, we conducted experiments on "large-sized" data sets, which we refer to as the "Non_normal (Large-size)" and "Adult" sets, which consisted of 20,000 patterns and 8 dimensions, and 33,330 samples and 14 dimensions respectively. In this case, because the size of the sets was reasonably large, the recursive version of the SVM was invoked to a depth of *four*.

As in the case of the "Non_normal (Medium-size)" data set, the data set "Non_normal (Large-size)" was generated randomly with the normal distributions. The "Adult" data set, which had been extracted from a census bureau database[1], has also been obtained from the UCI Machine Learning Repository [17]. The aim of the pattern recognition task here was to distinguish the income into two groups, in the first group the salary is more than 50K dollars, and in the second group the salary is less than or equal to 50K dollars. Each sample vector has fourteen attributes. Some of the attributes, such as the age, hours-per-week, etc., are continuous numerical values. The others, such as education, race, etc., are nominal symbols. In the experiments, the nominal attributes were replaced with numeric zeros.

[1] http://www.census.gov/ftp/pub/DES/www/welcome.html

Table 2. The experimental results of the recursive SVM for the "Adult" data set. Here, *Depth*(i) means the depth at which the data set is sub-divided into 2^{i-1} subsets. *DS*, *CT*, *SV* and *Acc* are the data set size (the number of sample vectors), the processing CPU-time (in seconds), the number of support vectors, and the classification accuracy rate (%), respectively

Depth(i)	*DS1*	*CT1*	*SV1*	*Acc1*	*DS2*	*CT2*	*SV2*	*Acc2*
1	16665	3825.46	6448	82.84	16665	2365.16	6218	82.63
4	2084	19.19	843		2084	17.07	801	
	2083	17.82	819		2083	15.04	814	
	2083	15.94	825		2083	14.01	771	
	2083	20.17	821		2083	19.32	825	
	2083	15.94	836		2083	29.01	810	
	2083	21.75	807		2083	12.10	802	
	2083	15.26	853		2083	17.01	808	
	2083	18.39	814		2083	16.20	775	
	6621	256.80	6270	81.28	6406	219.46	6059	79.49

Although the experimental results for the "Non_normal (Large-size)" and the "Adult" data sets are given in [16], we briefly cite the results for the latter in Table 2.

Consider Table 2. At the *depth* 1, the data set of 16,665 samples was processed requiring a computation time of 3,825.46 seconds, and gave an accuracy of 82.84 % with a reduction of 61.31 %. However, if the 16,665 samples are subdivided into eight subsets of 2,083 each at the *depth* 4, processing each of these involves only the times given in the third column whose average is 18.06 seconds, leading to 843, 819, 825, 821, 836, 807, 853 and 814 reduced prototypes, respectively. When *these* 6,621 samples are, in turn, subjected to a pure SVM method, the number of reduced samples reduced to 6,270 in 256.80 seconds and yielded an accuracy of 81.28 %. As the recursive computations were done serially, the time required was 401.26 seconds, which is only 10.5 % of the time which the original SVM would take. The power of the newly-introduced recursive philosophy is obvious!

5 Conclusions

Conventional PRS (Prototype Reduction Schemes) suffer from a major disadvantage, namely that of the excessive computational burden encountered by processing all the data, even though the sample data in the interior of the Voronoi space is typically processed for no reason. In this paper, we proposed a recursive mechanism, where the data sets are recursively sub-divided into smaller subsets, and the prototype points which are ineffective in the classification are eliminated for the subsequent invocations of the PRS. These prototypes are, in turn, gathered and processed by the PRS to yield more refined prototypes.

The proposed method was tested on both artificial and real-life benchmark data sets (of both medium and large sizes), and compared with the reported conventional methods, and the superiority has been clearly demonstrated both with regard to the required CPU time and the classification accuracy. The results obtained are conclusive and prove that it is futile to invoke a PRS on any large data set. Rather, it is expedient to recursively split the data, and invoke the PRS on the smaller subsets. Undoubtedly, the advantage of the recursive invocations increases with the size of the data set.

Acknowledgements

The work of the first author was done while visiting with the School of Computer Science, Carleton University, Ottawa, Canada. The work of the second author was partially supported by the NSERC of Canada. The authors are very grateful to Luis Rueda who helped us with the manuscript.

References

1. A. K. Jain, R. P. W. Duin and J. Mao.: Statistical pattern recognition: A review. *IEEE Trans. Pattern Anal. and Machine Intell.*, **PAMI-22(1)**:4–37, 2000. 528
2. D. V. Dasarachy.: *Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques*. IEEE Computer Society Press, Los Alamitos, 1991. 529, 530
3. P. E. Hart.: The condensed nearest neighbor rule. *IEEE Trans. Inform. Theory*, **IT-14**: 515–516, May 1968. 529, 530
4. G. W. Gates.: The reduced nearest neighbor rule. *IEEE Trans. Inform. Theory*, **IT-18**: 431–433, May 1972. 529
5. C. L. Chang.: Finding prototypes for nearest neighbor classifiers. *IEEE Trans. Computers*, **C-23(11)**: 1179–1184, Nov. 1974. 529, 530
6. G. L. Ritter, H. B. Woodruff, S. R. Lowry and T. L. Isenhour.: An algorithm for a selective nearest neighbor rule. *IEEE Trans. Inform. Theory*, **IT-21**: 665–669, Nov. 1975. 529
7. I. Tomek.: Two modifcations of CNN. *IEEE Trans. Syst., Man and Cybern.*, **SMC-6(6)**: 769–772, Nov. 1976. 529
8. P. A. Devijver and J. Kittler.: On the edited nearest neighbor rule. *Proc. 5th Int. Conf. on Pattern Recognition*, 72–80, Dec. 1980. 529
9. K. Fukunaga.: *Introdction to Statistical Pattern Recognition, Second Edition*. Academic Press, San Diego, 1990. 529, 532
10. Q. Xie, C. A. Laszlo and R. K. Ward.: Vector quantization techniques for nonparametric classifier design. *IEEE Trans. Pattern Anal. and Machine Intell.*, **PAMI-15(12)**: 1326–1330, Dec. 1993. 529, 532
11. Y. Hamamoto, S. Uchimura and S. Tomita.: A bootstrap technique for nearest neighbor classifier design. *IEEE Trans. Pattern Anal. and Machine Intell.*, **PAMI-19(1)**:73–79, Jan. 1997. 529, 532
12. C. J. C. Burges.: A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, **2(2)**:121–167, 1998. 529, 530
13. S.-W. Kim and B. J. Oommen.: Enhancing prototype reduction schemes with LVQ3-type algorithms. To appear in *Pattern Recognition*. 529, 530

14. T. Kohonen.: *Self-Oganizing Maps*. Berlin, Springer - Verlag, 1995. 530
15. N. Aras, B. J. Oommen and I. K. Altinel.: The Kohonen network incorporating explicit statistics and its application to the travelling salesman problem. *Neural Networks*, 1273–1284, Dec. 1999. 530
16. S.-W. Kim and B. J. Oommen.: Recursive prototype reduction schemes applicable for large data sets. *Unabridged version of this paper*. 530, 531, 533, 534, 535
17. http://www.ics.uci.edu/mlearn/MLRepository.html. 533, 534